Computation of Buffer Capacities for Throughput Constrained and Data Dependent Inter-Task Communication

Maarten H. Wiggers [1], Marco J. G. Bekooij [2] and Gerard J. M. Smit [1]
[1] University of Twente, Enschede, The Netherlands
[2] NXP Semiconductors, Eindhoven, The Netherlands
m.h.wiggers@utwente.nl

Abstract - *Streaming applications are often implemented as task graphs. Currently, techniques exist to derive buffer capacities that guarantee satisfaction of a throughput constraint for task graphs in which the inter-task communication is data-independent, i.e. the amount of data produced and consumed is independent of the data values in the processed stream. This paper presents a technique to compute buffer capacities that satisfy a throughput constraint for task graphs with data dependent inter-task communication, given that the task graph is a chain. We demonstrate the applicability of the approach by computing buffer capacities for an MP3 playback application, of which the MP3 decoder has a variable consumption rate. We are not aware of alternative approaches to compute buffer capacities that guarantee satisfaction of the throughput constraint for this application.*

1. Introduction

Many embedded systems process streams of data, such as smart-phones and set-top boxes that can process audio and video streams. In order to satisfy constraints on power dissipation and performance, multi-processor architectures are often the architecture of choice in these embedded systems. In this context, the processing of data streams often has end-to-end temporal constraints on latency and throughput, of which throughput constraints dominate for audio and video stream processing. These streaming applications are often implemented as task graphs, where tasks communicate data over First-In First-Out (FIFO) buffers implemented as circular buffers [8]. Each task execution only starts when there is sufficient data in the input buffers and sufficient space in the output buffers such that this execution can run to completion without further blocking. This execution condition is a robust mechanism to prevent buffer overflow.

Currently, techniques are available to provide throughput guarantees for applications in which each task has an execution condition that is data independent [10, 11, 14]. However, for instance audio and video encoders and decoders often have tasks with data dependent execution conditions that can change every execution, e.g. a variable length decoding task, for which these techniques are not applicable. For example, in Figure 1, we have a task graph with tasks w_a and w_b that communicate over a buffer. In this task graph, task w_a produces 3 data items in every execution and task w_b consumes either two or three data items in every execution. In case the consumption quantum equals three in every task execution, then the minimum buffer capacity for deadlock-free execution is three. However, if the consumption quantum equals two in every task execution, then the minimum buffer capacity for deadlock-free execution is four. This example shows that maximising the consumption quantum does not lead to buffer capacities that are sufficient for other consumption quanta.

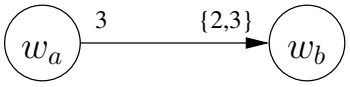

Figure 1. Task graph example

The contribution of this paper is an algorithm that computes buffer capacities for a class of task graphs with data dependent execution conditions, with the guarantee that these buffer capacities are sufficient to satisfy a given throughput constraint.

We assume that all shared resources have run-time arbiters. Further, we assume that we know upper and lower bounds on the amount of data and space that is required for task executions, and that we know an upper bound on the execution times of the tasks. Furthermore, we restrict ourselves to chains of tasks.

We present a dataflow model that allows every task execution to have a different execution condition. The approach to derive buffer capacities is to define linear bounds on the consumption and production times of data and space for each buffer. Subsequently, we show that for every sequence of consumption and production quanta there exists a schedule in which the consumption times are bounded from below by the linear bound on consumption times and the production times are bounded from above by the linear bound on production times. Buffer capacities are then derived with the difference between the lower bound on space

978-3-9810801-3-1/DATE08 © 2008 EDAA

consumption times and the upper bound on space production times, and the production and consumption rates. We will show that these buffer capacities are sufficient to satisfy the throughput constraint if other productions and consumption rates of the throughput determining task lead to reduced execution rates of the other tasks.

The outline of this paper is as follows. We first discuss related work in Section 2. Then in Section 3, we introduce our task model, our analysis model, and the relation between these two models. In Section 4, we present our approach to compute buffer capacities, which is applied to an MP3 playback application in Section 5 after which we conclude in Section 6.

2. Related Work

We see two classes of related work: (1) work that applies quasi static-order scheduling of tasks, and (2) work that applies run-time scheduling of tasks.

Quasi static-order schedules are constructed at design time and try to decide on most scheduling decisions at design time but may contain code that performs some data-dependent computations at run-time [1]. Work that applies quasi static-order scheduling includes parameterised dataflow [1], interval-limited dataflow [12], heterochronous dataflow [3], and hierarchical reconfiguration of dataflow models [7]. These approaches are all based on the notion of (sub)graph iterations, i.e. in these approaches a schedule is only changed at the end of an iteration, thereby limiting the frequency with which tasks can change their productions and consumptions. For boolean dataflow [2], a class of graphs is identified for which quasi static-order schedules can be constructed. However, it is in general undecidable whether a given boolean dataflow graph is part of this class.

The reason to restrict changes to the dataflow graph to the end of (sub)graph iterations is that when applying quasi static-order scheduling one needs to construct a schedule, which requires the existence of a (parameterised) bounded length schedule. Therefore the advantage of applying run-time scheduling is that instead of the requirement to construct a schedule at design time that is applicable for all sequences of productions and consumptions, we only need to show that for all sequences of productions and consumptions a schedule exists at run-time.

Run-time arbitration is applied in real-time calculus [6] and Symta/S [4]. However, both approaches do not allow cyclic dependencies in the graph that influence the temporal behaviour, such as for instance caused by a data producing task that only starts when there is sufficient space. The implication is that these approaches are only applicable if, for every task, there is a fixed number of executions over which the consumption is constant, as for instance in a constant bit-rate decoder. Otherwise the consuming task needs to slow down the producing task. For instance, if, in the task graph of Figure 1, the application requires that task w_b executes with a certain period, then task w_a needs to be able to keep up with the situation in which in every execution of w_b the consumption is three. However, when in every execution of w_b the consumption is two, then for any non-terminating schedule a finite buffer capacity only exists if the execution rate of task w_a is reduced. Cyclo-dynamic [13] and bounded dynamic dataflow [9] also apply run-time scheduling, but do not provide algorithms to compute buffer capacities that guarantee the existence of a non-terminating schedule, let alone satisfy a throughput constraint.

Techniques that rely on static task dependencies [10] or on the existence of a periodic schedule [11] cannot be applied, because the task dependencies are dynamic and there is no periodic schedule if productions and consumptions can vary in every execution.

In contrast with related work, we present an algorithm to compute buffer capacities that are sufficient to satisfy a throughput constraint, while allowing the amount of data that is consumed and produced to change in every execution of the tasks. The presented approach builds upon the technique presented in [14], in which buffer capacities are computed for applications with data-independent inter-task communication. The fact that the approach in this paper deals with data-dependent communication is reflected in the following two ways. The first aspect is that the difference between the lower bound on consumption times and the upper bound on production times is increased to allow the existence of a schedule for any sequence of consumptions and productions. The second aspect is that with data-dependent consumptions and productions the schedule that will occur at run-time can be delayed compared to the schedule shown to exist when computing the buffer capacities. The latter for instance occurs for the task graph shown in Figure 1, where task w_b can reduce the execution rate of task w_a.

3. Task and Analysis Models

In this section, we first introduce our task model, then define the analysis model, and conclude by presenting the relation between these two models.

3.1. Task Model

We assume that an application is implemented as a task graph. A task graph is a weakly connected directed graph $T = (W, B, \xi, \lambda, \kappa, \zeta)$. A weakly connected directed graph is a graph for which the underlying undirected graph is connected. We restrict the topology of task graphs to chains, i.e. for every task we have that the number of input buffers is maximally one and also that the number of output buffers is maximally one. Furthermore, we require that the throughput requirement is either on a task that does not have any output buffers or on a task that does not have any input buffers. In such a task graph we have that tasks w_a and w_b, with $w_a, w_b \in W$, can communicate over a circular buffer $b_{ab} \in B$. Let a buffer b_{ab} denote a buffer over which task w_a sends data to task w_b. We say that tasks consume and produce containers on these circular buffers, where a container is a place-holder for data and all containers in a

buffer have a fixed size. Tasks only start an execution when the previous execution has finished and there are sufficient full containers on their input buffers and sufficient empty containers on their output buffers such that the execution can finish without further waiting on container arrivals. The number of full containers that a task w_b requires on buffer $b_{ab} \in B$ is a value from $\lambda(b_{ab})$, with $\lambda : B \rightarrow \mathcal{P}_f(\mathbb{N})$. We let $\mathbb{N}$ denote the set of non-negative integer values, and we let $\mathcal{P}_f(\mathbb{N})$ denote the set of all finite subsets of $\mathbb{N}$ excluding the empty subset and the set only consisting of the value zero. The number of full containers that a task w_a produces on buffer $b_{ab} \in B$, which equals the number of empty containers that are required, is a value from $\xi(b_{ab})$, with $\xi : B \rightarrow \mathcal{P}_f(\mathbb{N})$. The worst-case response time is defined as the maximum difference between the time at which sufficient containers are present to enable an execution of task w_a and the time at which this execution finishes. The worst-case response time of task w_a is denoted by $\kappa(w_a)$, with $\kappa : W \rightarrow \mathbb{R}^+$. As in [15], we allow tasks to be scheduled at run-time by arbiters that can guarantee a worst-case response time given the worst-case execution times and the scheduler settings, i.e. the guarantee is independent of the rate with which tasks start their execution. This class of schedulers, for instance, includes time-division multiplex and round-robin. The capacity of a circular buffer b is given by $\zeta(b)$, with $\zeta : B \rightarrow \mathbb{N}$. We require that every buffer is initially empty.

3.2. Analysis Model

We call our analysis model Variable-Rate Dataflow (VRDF). A VRDF graph $G = (V, E, \pi, \gamma, \delta, \rho)$ is a directed graph that consists of a finite set of actors V and a finite set of edges E. A firing of an actor is enabled when on all input edges of the actor sufficient tokens are present. The number of tokens that are required on an edge $e \in E$ in a particular firing is the token consumption quantum in that firing on edge e, which is a value taken from $\gamma(e)$. We define $\gamma : E \rightarrow \mathcal{P}_f(\mathbb{N})$, with furthermore $\hat{\gamma}(e) = \max(\gamma(e))$ denoting the maximum token consumption quantum and $\check{\gamma}(e) = \min(\gamma(e))$ denoting the minimum token consumption quantum. The token production quantum on edge e is a value from $\pi(e)$. We define $\pi : E \rightarrow \mathcal{P}_f(\mathbb{N})$, with furthermore $\hat{\pi}(e) = \max(\pi(e))$ denoting the maximum token production quantum on edge e and $\check{\pi}(e) = \min(\pi(e))$ denoting the minimum token production quantum on edge e. The number of initial tokens on edge e is given by $\delta(e)$, with $\delta : E \rightarrow \mathbb{N}$, while the response time of an actor $v \in V$ is given by $\rho(v)$, with $\rho : V \rightarrow \mathbb{R}^+$. An actor v consumes its tokens in an atomic action at the start of a firing and produces its tokens in an atomic action $\rho(v)$ later at the finish of the firing. Actors do not start a firing before every previous firing has finished.

Definition 1 (Monotonic execution in the start times)
A dataflow graph executes monotonically in the start times if no decrease Δ in the start time of any firing can lead to an increase in the start time of any other firing.

A VRDF graph executes monotonically in the start times. This is because the firing rules and token production rules of a firing are independent of the start time of the firing. Therefore, if a firing starts earlier, then tokens will only be produced earlier, which only leads to an earlier enabling and start of other actors.

Definition 2 (Linear execution in the start times)
A dataflow graph has linear temporal behaviour if a delay Δ in the start times cannot lead to delay larger than Δ for any start time of any firing.

A VRDF graph has linear temporal behaviour, because a start time that is delayed by Δ can only lead to token productions that are delayed by maximally Δ. These tokens can delay another start time by maximally Δ, and so on.

The functional behaviour of VRDF graphs is deterministic in the sense that it is schedule independent, because the production and consumption quanta are selected independently of the token arrival times.

3.3. Construction of Analysis Model

We construct a VRDF graph $G = (V, E, \pi, \gamma, \delta, \rho)$ from a task graph $T = (W, B, \xi, \lambda, \kappa, \zeta)$ as follows. Every task $w \in W$ is modelled by an actor $v \in V$, where the response time of the actor equals the worst-case response time of the task, i.e. $\rho(v) = \kappa(w)$. A buffer $b_{ab} \in B$ from task w_a to task w_b is modelled by two edges in opposite direction between the actors that model the tasks, i.e. edges $e_{ab}, e_{ba} \in E$ are added if v_a models w_a and v_b models w_b. The capacity of buffer b_{ab} is modelled by initial tokens on edge e_{ba}, i.e. $\delta(e_{ba}) = \zeta(b_{ab})$. The number of containers produced on buffer b_{ab} is modelled by an equal number of tokens produced on edge e_{ab}, i.e. $\pi(e_{ab}) = \xi(b_{ab})$, and an equal number of tokens consumed from edge e_{ba}, i.e. $\gamma(e_{ba}) = \xi(b_{ab})$. The number of tokens consumed per firing from edge e_{ba} models the number of empty containers that are required for task w_a to start. Similarly, we have that $\gamma(e_{ab}) = \lambda(b_{ab})$ and $\pi(e_{ba}) = \lambda(b_{ab})$.

We know that in every execution, a task requires on its output buffer as many empty containers as the number of containers it produces, and in every execution the task produces on its input buffer as many empty containers as were consumed. Since we furthermore restrict ourselves to task graphs that are chains, we have that the VRDF graph is inherently strongly consistent [1, 5].

4. Buffer Capacity Computation

This section starts by presenting the basic idea behind our approach to compute buffer capacities. Subsequently, we present our approach to compute buffer capacities that guarantee satisfaction of a throughput constraint for producer-consumer pairs. This section is concluded by showing how buffer capacities can be computed for task graphs that are chains.

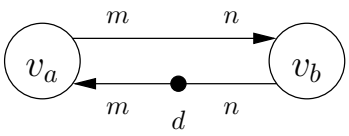

Figure 2. Example VRDF graph.

4.1. Basic Idea

In this section, we will provide the basic idea of our approach using the VRDF graph shown in Figure 2 that models the task graph, as shown in Figure 1, with $m = \{3\}$ and $n = \{2, 3\}$. In a later section, we will show that the problem of deriving buffer capacities in a chain can be reduced to multiple instances of the problem of deriving buffer capacities for a produced-consumer pair. We assume that the application specifies the throughput constraint that actor v_b should execute strictly periodically with period τ. With this throughput constraint, we compute the maximum production and consumption rates of each actor on each edge, as required to guarantee that sufficient tokens are always available for periodic execution of actor v_b. In this example, we have that on edge e_{ab} the maximum consumption rate of actor v_b is three tokens per τ, which means that the maximum required production rate of actor v_a is also three tokens per τ. Similarly on edge e_{ba}, the maximum required consumption rate of actor v_a is also three tokens per τ.

Figure 3 shows two firings of actor v_b. The filled dots denote the production times of tokens, while the open dots denote the consumption times of tokens. In this example, actor v_b has a sequence of productions and consumptions in which it first consumes and produces two tokens and in the next firing consumes and produces three tokens.

We now construct bounds on the consumption and production times that have the maximum required rates and are such that for every sequence of productions and consumptions, a schedule exists such that these bounds are conservative. In Figure 3, the upper bound on token production times is denoted $\hat{\alpha}_p$ and the lower bound on token consumption times is denoted $\check{\alpha}_c$.

Subsequently, we use the lower bound on token consumption times of actor v_a on edge e_{ba} and the upper bound on token production times of actor v_b on edge e_{ba} to derive the buffer capacity. For actor v_a, we require that for every sequence of token transfers there exists a valid schedule such that the bounds are conservative. The main novelty in our approach is that we show that the buffer capacities are still sufficient if we delay the schedule of actor v_b, which we have shown to exist, in order to execute with the required period τ.

4.2. Buffer Capacities for Producer-Consumer Pairs

In this section, we initially assume the VRDF graph G as given in Figure 2, i.e. with unspecified m and n. For this producer-consumer pair, we will come up with an expression for the buffer capacity that is sufficient to satisfy

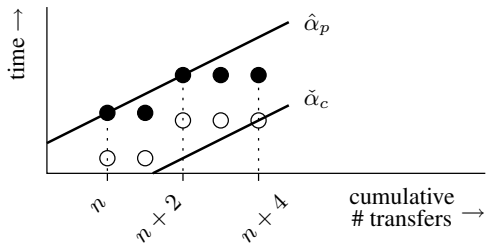

Figure 3. Example schedule of actor v_b, showing when a particular number of tokens is consumed and produced.

the throughput constraint. At the end of this section, we will show that the computation of buffer capacities for a task graph that is a chain can be decomposed in computing buffer capacities of producer-consumer pairs.

We again assume that actor v_b is required to execute strictly periodically with period τ. This means that actor v_b can consume with a rate of $\frac{\tau}{\hat{\gamma}(e_{ab})}$ from edge $e_{ab} \in E$ and can produce with a rate of $\frac{\tau}{\hat{\pi}(e_{ba})}$ on edge $e_{ba} \in E$. In order to let v_b execute strictly periodically with period τ, while only requiring a finite number of initial tokens on edge e_{ba}, actor v_a needs to be able to consume with a rate $\frac{\tau}{\hat{\pi}(e_{ba})}$ from edge e_{ba} and produce with a rate of $\frac{\tau}{\hat{\gamma}(e_{ab})}$ on edge e_{ab}.

Construction of Bounds Given two edges e_{ab} and e_{ba} that together model a buffer. Further given a linear upper bound on token production times, $\hat{\alpha}_p$, on edge e_{ab}. Then we can show that for every sequence of token transfer quanta there is a schedule of which the token consumption times allow for a linear lower bound on token consumption times, $\check{\alpha}_c$, that has a bounded difference with $\hat{\alpha}_p$. This schedule is such that for instance for v_a, we have that any firing that produces tokens x to $x + m - 1$ produces token x at the time that is specified by the upper bound on token production times. See for instance the upper bound on production times in Figure 4 and the schedule that is such that this bound just remains conservative.

Given this schedule, we can derive the minimum difference between the linear upper bound on production times on edge e_{ab}, $\hat{\alpha}_p$ in Figure 4, and the linear lower bound on consumption times on edge e_{ba}, $\check{\alpha}_c$ in Figure 4. This can be done by realising that in every firing the consumption time is $\rho(v_a)$ earlier than the production time. Further we have that while the upper bound on token productions needs to bound the production time of token x, the lower bound on token consumptions needs to bound the consumption time of token $x + m - 1$. This can be seen in Figure 4, where the linear lower bound on token consumptions allows for a consumption quanta of $\hat{m} = \hat{\gamma}(e_{ba}) = \hat{\pi}(e_{ab}) = 3$ in every firing. The minimum difference between both bounds of actor v_a that is sufficient to allow for a schedule to exist with token production and consumption times which for

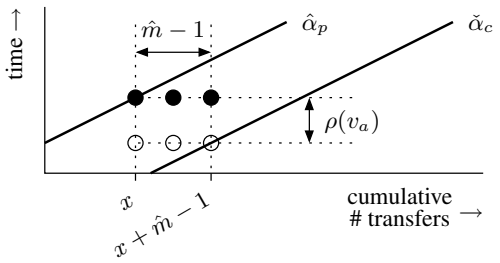

Figure 4. Derivation of difference between token transfer bounds

every sequence of token transfer quanta are conservatively bounded is therefore

$$\hat{\alpha}_p(e_{ab}) - \check{\alpha}_c(e_{ba}) = \rho(v_a) + \frac{\tau}{\hat{\pi}(e_{ba})} \cdot (\hat{\gamma}(e_{ba}) - 1) \quad (1)$$

Similarly for actor v_b this difference equals

$$\hat{\alpha}_p(e_{ba}) - \check{\alpha}_c(e_{ab}) = \rho(v_b) + \frac{\tau}{\hat{\gamma}(e_{ab})} \cdot (\hat{\gamma}(e_{ab}) - 1) \quad (2)$$

Sufficient Initial Tokens An actor is only enabled when sufficient tokens are present. For the VRDF graph shown in Figure 2 this means that the consumption time of token x by actor v_b should not be earlier than the production time of token x by actor v_a. Since we have linear bounds on the consumption and production times, this means that for any token $\hat{\alpha}_p(e_{ab}) \leq \check{\alpha}_c(e_{ab})$. With the knowledge of Equations (1) and (2), this creates a minimum difference between the bound on productions on edge e_{ba} and the bound on consumptions on edge e_{ba} of

$$\hat{\alpha}_p(e_{ba}) - \check{\alpha}_c(e_{ba}) = \rho(v_a) + \rho(v_b) + \frac{\tau}{\hat{\pi}(e_{ba})} \cdot (\hat{\gamma}(e_{ba}) - 1) + \frac{\tau}{\hat{\gamma}(e_{ab})} \cdot (\hat{\gamma}(e_{ab}) - 1) \quad (3)$$

This difference in production and consumption times together with the production and consumption rates translates to a sufficient number of initial tokens. Since $\hat{\pi}(e_{ba}) = \hat{\gamma}(e_{ab})$, we have that $\frac{\tau}{\hat{\pi}(e_{ba})} = \frac{\tau}{\hat{\gamma}(e_{ab})}$. Further, we have that the rate of bounds $\hat{\alpha}_p(e_{ba})$ and $\check{\alpha}_c(e_{ba})$ equals $\tau/\hat{\pi}(e_{ba})$. The horizontal difference between bounds $\hat{\alpha}_p(e_{ba})$ and $\check{\alpha}_c(e_{ba})$ is therefore $\hat{\pi}(e_{ba})/\tau \cdot (\hat{\alpha}_p(e_{ba}) - \check{\alpha}_c(e_{ba}))$. Since tokens are counted starting from 1, we have that according to the linear bounds

$$\frac{\hat{\pi}(e_{ba})}{\tau} \cdot (\hat{\alpha}_p(e_{ba}) - \check{\alpha}_c(e_{ba})) + 1 \quad (4)$$

tokens are consumed before the first token is produced on edge e_{ba}.

A number of initial tokens that equals the largest integer smaller than or equal to Equation (4) is therefore sufficient to ensure that sufficient tokens are available to allow for the schedules that have token production and consumption times that are conservatively bounded by the derived linear bounds.

Producer Schedule A schedule for actor v_a is not valid if the difference between subsequent starts is less than the response time of actor v_a. In our schedule we have that the minimum difference between subsequent starts occurs for the minimum token production quantum, and equals $\check{\pi}(e_{ab}) \cdot \frac{\tau}{\hat{\gamma}(e_{ab})}$. Therefore iff $\rho(v_a) \leq \check{\pi}(e_{ab}) \cdot \frac{\tau}{\hat{\gamma}(e_{ab})}$, then for every sequence of token transfer quanta a valid schedule exists for v_a such that the linear bounds are conservative.

Note that in the schedule that we have just shown to exist, we are allowed to have a larger difference between subsequent starts than will occur when the task graph executes. This is because we have a one-to-one relation between the VRDF graph and the task graph and we know that an VRDF graph executes monotonically in the start times.

Consumer Schedule For actor v_b, we also have that a schedule is not valid if the difference between subsequent starts is less than the response time of actor v_b. This requirement means that these schedules are only valid if $\rho(v_b) \leq \pi(e_{ba}) \cdot \frac{\tau}{\hat{\pi}(e_{ba})}$, which is only the case if $\pi(e_{ba}) = \hat{\pi}(e_{ba})$ in every execution. For any other sequence of token transfer quanta by v_b, i.e. in which $\pi(e_{ba}) < \hat{\pi}(e_{ba})$, the constraint that a firing does not start before every previous firing has finished will lead to a delay Δ, with $\Delta > 0$, of the start time of v_b compared to the schedule that has token production and consumption times on e_{ba} and e_{ab}, which we have conservatively bounded. However, since the graph has linear temporal behaviour we know that a delay of a start time by Δ cannot lead to a delay of more than Δ of any firing of any actor. Therefore if tokens arrive in time to enable the schedules that have token production and consumption times that we have conservatively bounded, then tokens will also arrive in time to enable the schedule of v_b that is periodic with period τ in case $\pi(e_{ba}) < \hat{\pi}(e_{ba})$.

Note that we allow the situation in which actor v_b has firings in which it does not consume any tokens from particular edges. Traditionally, this situation is not allowed, because in this case no solution exists for the balance equations, which means that no quasi-static schedule can be computed.

4.3. Extension to Chains

In case we have a task graph that is a chain, buffer capacities can be computed per producer-consumer pair of tasks as follows. In Equation (4), we have that the only parameters that depend on the topology of the graph are the rates of the bounds. Let actor v_τ model the task with no output buffers of which the application requires that it executes strictly periodically. Then on every buffer, we have that the data consuming task determines the rate, which means that on each buffer the data producing task should have a minimum production rate that is at least equal to the maximum consumption rate of the data consuming task. Consider a producer-consumer pair with data producing task w_x and data consuming task w_y, which is modelled by actors

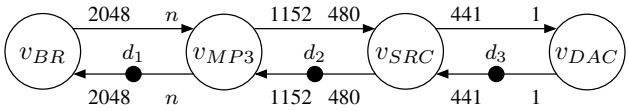

Figure 5. VRDF graph of MP3 application.

v_x and v_y. Let $\phi(v_y)$ be the minimal required difference between subsequent starts of actor v_y, which is τ in case $v_y = v_\tau$. Then the rate of the bounds for this producer-consumer pair is $\phi(v_y)/\hat{\gamma}(e_{xy})$. From this it follows that $\phi(v_x) = (\phi(v_y)/\hat{\gamma}(e_{xy})) \cdot \check{\pi}(e_{xy})$, which can be used to compute the rate of the bounds on the buffer from which w_x consumes data.

4.4. Throughput Constraint on Source

If, in the VRDF graph shown in Figure 2, instead of v_b we have that v_a is required to execute strictly periodically with period τ, then the presented approach needs to be adapted as follows. First of all, the rate of the bounds should now reflect the maximum rate with which v_a can consume and produce, which equals $\tau/\hat{\pi}(e_{ab}) = \tau/\hat{\gamma}(e_{ba})$. This implies that $\hat{\pi}(e_{ba})$ is replaced by $\hat{\pi}(e_{ab})$ in Equation 4. Further instead of the constraint on the response time of the producer, we now have a constraint on the response time of the consumer. Similarly, instead of allowing n to attain the value zero we now allow m to attain the value zero. The extension to a chain has a similar difference. Instead of maximising consumption and minimising production in order to derive the maximum execution rate relative to the sink, we now need to maximise production and minimise consumption to obtain the maximum execution rate of each actor relative to the actor that models the source of the task graph.

5. Experimental Results

In this section, we apply our approach to an MP3 playback application for a variable bit-rate stream with a sample-rate of 48 kHz. The VRDF graph of this application is shown in Figure 5. In this graph we have an actor v_{BR} that reads blocks of bytes from a compact disc, an actor v_{MP3} that decodes the compressed audio, a sample-rate converter, v_{SRC}, from 48 kHz to 44.1 kHz and a digital-to-analog converter v_{DAC}. The application requires that v_{DAC} executes strictly periodically with a frequency of 44.1 kHz. Given that an MP3 frame contains 1152 samples and given a sampling frequency of 48 kHz, we have that with a maximum bit-rate of 320 kbit per second the maximum number of bytes per frame equals 960. From the throughput constraint, we can derive response times that would just allow the throughput constraint to be satisfied. These are $\rho(v_{BR})$ = 51.2 ms, $\rho(v_{MP3})$ = 24 ms, $\rho(v_{SRC})$ = 10 ms, and $\rho(v_{DAC})$ = 0.0227 ms.

With these response times, we require the following number of initial tokens $d_1 = 6015$, $d_2 = 3263$, and $d_3 = 882$. With our dataflow simulator we have verified that these buffer capacities are indeed sufficient to satisfy the throughput constraint. We obtain a lower bound on the required buffer capacities by assuming that n is constant and equals 960. Using traditional analysis techniques [10], we obtain $d_1 = 5888$, $d_2 = 3072$, and $d_3 = 882$. The difference occurs, because our approach accounts for the variation in quanta, and uses linear bounds to derive buffer capacities.

6. Conclusion

We have presented an approach to compute buffer capacities that satisfy a throughput constraint. In contrast with existing approaches this approach can also be applied when the number of consumptions and productions of tasks are data-dependent and can vary from execution to execution.

An important difference with current approaches that use dataflow models is that we apply run-time arbitration in our system, which means that we do not need to construct a schedule, but only need to show the existence of a schedule. Current approaches that apply run-time arbitration have difficulties with the analysis of systems in which the start of a task is dependent on the amount of space in its output buffers. However, if productions and consumptions can change every execution then such a dependency is unavoidable to prevent buffer overflow.

We expect that the concept of linear temporal behaviour as presented in this paper will allow us to extend our current approach to VRDF graphs of any topology. For these graphs, it is no longer possible to consider producer-consumer pairs, but instead it is required to consider all paths between actors when showing the existence of schedules. The extension to VRDF graphs of any topology would result in an important extension of the class of applications for which guarantees on their temporal behaviour can be provided.

References

[1] B. Bhattacharya and S. S. Bhattacharyya. Parameterized Dataflow Modeling for DSP Systems. *IEEE Transactions on Signal Processing*, 49(10), 2001.

[2] J. Buck. *Scheduling Dynamic Dataflow Graphs with Bounded Memory using the Token Flow Model*. PhD thesis, University of California at Berkeley, 1993.

[3] A. Girault *et al.* Hierarchical Finite State Machines with Multiple Concurrency Models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6), 1999.

[4] M. Jersak *et al.* Performance Analysis of Complex Embedded Systems. *International Journal of Embedded Systems*, 1(1-2), 2005.

[5] E. A. Lee. Consistency in Dataflow Graphs. *IEEE Transactions on Parallel and Distributed Systems*, 2(2), 1991.

[6] A. Maxiaguine *et al.* Tuning SoC Platforms for Multimedia Processing: Identifying Limits and Tradeoffs. In *Proc. CODES+ISSS*, 2004.

[7] S. Neuendorffer and E. A. Lee. Hierarchical Reconfiguration of Dataflow Models. In *Proc. MEMOCODE*, 2004.

[8] A. Nieuwland *et al.* C-HEAP: A Heterogeneous Multi-Processor Architecture Template and Scalable and Flexible Protocol for the Design of Embedded Signal Processing Systems. *Design Automation for Embedded Systems*, 7(3), 2002.

[9] M. Pankert *et al.* Dynamic Data Flow and Control Flow in High Level DSP Code Synthesis. In *Proc. Int'l Conference on Acoustics, Speech, and Signal Processing*, 1994.

[10] S. Sriram and S.S. Bhattacharyya. *Embedded Multiprocessors: Scheduling and Synchronization*. Marcel Dekker Inc., 2000.

[11] S. Stuijk *et al.* Exploring Trade-Offs in Buffer Requirements and Throughput Constraints for Synchronous Dataflow Graphs. In *Proc. DAC*, 2006.

[12] J. Teich and S. S. Bhattacharyya. Analysis of Dataflow Programs with Interval-Limited Data-Rates. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 43(2-3), 2006.

[13] P. Wauters *et al.* Cyclo-Dynamic Dataflow. In *Proc. Workshop on Parallel and Distributed Processing*, 1996.

[14] M. H. Wiggers *et al.* Efficient Computation of Buffer Capacities for Multi-Rate Real-Time Systems with Back-Pressure. In *Proc. CODES+ISSS*, 2006.

[15] M. H. Wiggers *et al.* Efficient Computation of Buffer Capacities for Cyclo-Static Real-Time Systems with Back-Pressure. In *Proc. RTAS*, 2007.