Predictability and Utilisation Trade-off in the Dynamic Management of Multiple Video Stream Decoding on Network-on-Chip based Homogeneous Embedded Multi-cores

Hashan Roshantha Mendis
Real-time Systems Group
Department of Computer Science
University of York
hrm506@york.ac.uk

Leandro Soares Indrusiak
Real-time Systems Group
Department of Computer Science
University of York
lsi@cs.york.ac.uk

Neil C. Audsley
Real-time Systems Group
Department of Computer Science
University of York
neil.audsley@york.ac.uk

ABSTRACT

Guaranteed admission control decisions in embedded multi-core platforms often cause a trade-off between predictability for utilisation. The state-of-the art shows that both these objectives cannot be optimised if the workloads are dynamic and are not known a priori. Deterministic admission control approaches use worst-case response-time calculations of the tasks and flows live in the system to analytically make safe admission decisions. These tests often result in significantly under-utilised systems. Using a poor admission controller may improve system utilisation but at the cost of low-predictability. In a soft-real time system under heavy system load it is acceptable to have a few deadline misses in order to maintain relatively higher utilisation levels. This paper presents heuristic based admission control test that attempts to find a compromise between predictability and utilisation. The heuristic adjusts the estimation of subtask deadline assignment proportionally, and uses this to estimate the lateness of tasks that are admitted and live in the system. We explore the performance consequences of the proposed heuristic-based approach using an abstract simulator, and analyse its effectiveness against a deterministic admission control test under low and heavy workloads.

1. INTRODUCTION

Complex multimedia activities (such as decoding multiple simultaneous video streams on a single device) require soft real-time support due to their sensitivity to delay and jitter. The dynamic nature of video streaming is challenging for the platforms as it is difficult to provide reasonable predictability guarantees whilst utilising the platform resources, when admitting video streams into the system. The state-of the art in predictable admission control of dynamic workloads uses deterministic approaches that utilise worst-case latencies to guarantee the workloads can be serviced within their timing constrains. However these methods often result in considerably under-utilised system resources.

The intent of this paper is to explore the possibility of using a thresholded-heuristic based approach to enable admission of dynamic video streams whilst improving system utilisation and service predictability. An MPEG (Moving Picture Experts Group) decoding task has a soft deadline proportional to the video frame-rate, and missing this deadline or dropping tasks due to over-utilisation will cause the video image frame to distort or freeze and will result in viewer dissatisfaction. Buffering has been employed on the client-side in many real-time video streaming applications when the throughput requirements of the video processing application is not satisfied. However if the waiting time for the buffering process is too long it may cause a negative quality of experience (QoE) [9]. Further, videos freezing/stalling mid-way during the playback of a stream may impact the QoE factor and even cause lower user-engagement. Thus guaranteeing a reliable video streaming service at the start of the stream is critical to improving user-experience [7].

In a modern house-hold, streaming a video content from the internet may require computational and communicational data redundancy due to the multiple multimedia devices (e.g. tablets, smart-phones, digital set-top boxes etc.). As each of these multimedia devices often consumes a high level of network bandwidth, overall network bandwidth usage may be affected negatively. One way to address this issue is to utilise a low-power embedded media server that acts as a central gateway to the media, performing parallel streaming, decoding and transcoding functions to multiple client devices. The embedded media gateway for this solution should support several concurrent video processing requests and should guarantee the required quality of service (QoS) levels of each request, whilst not degrading the QoS levels of any active video streams. The motivation of this research is driven from an industrial case-study to develop the mentioned media gateway that supports several concurrent video streams whilst guaranteeing the required high QoS for each multimedia device.

The remainder of this paper is organised as follows: Sec-

RTNS 2014 , October 8 - 10 2014, Versailles, France

http://dx.doi.org/10.1145/2659787.2659826.

tion 2 discusses the background of this research. Section 3 explains the platform and application model used by the proposed algorithm. Section 4, outlines the problem statement and define our objectives. Section 5 describes our heuristic based admission control test and outline our baselines. In Sections 6 and 7 we describe the chosen evaluation method and the experimental results.

2. BACKGROUND RESEARCH

This research work falls under several research domains, and hence this section has been separated into the following categories: video processing hardware, multi-stream video processing, deterministic vs. statistical admission control and feedback based admission control.

Processing of high-definition video streams, specifically for real-time applications, has been a challenging task due to the complexity of the algorithms performed on high-resolution images. In addition, these tasks require considerable amounts of memory bandwidth. Traditionally, the compute-intensive tasks of video processing have been implemented on Field Programmable Gate Array (FPGA) based dedicated hardware accelerators due to their high customisability and performance improvements [3]. Recently, graphics processing units (GPUs) have also offered a viable solution to accelerate the various numerical and signal processing algorithms required by video processing tasks. Modern GPUs may consist of up to hundreds of processing cores capable of exploiting the parallelism in these video processing algorithms to meet the required application throughput [4]. Both GPUs and FPGA technologies are viable approaches to implement a multi-stream video processing platform, however their high cost and power requirements are potential concerns if they are to be used as a embedded consumer device.

Ditze et al. [6] proposes a method for real-time scheduling and admission control of multiple MPEG video streams which uses continuous re-processing by the admission controller and scheduling the different workloads out-of-phase to each other such that the decoding tasks do not interfere. In case the allocated resources vary over time, their method re-invokes the admission controller to adjust the resource reservation. This method guarantees quality of service, however their analysis is limited to a few concurrent video streams and a single-core system. Blanch et al. [2] presents several scheduling strategies and task assignment metrics used in decoding multiple MPEG-4 video streams on a heterogeneous multi-core platform. Selecting a task from the multiple video streams is developed based on estimated task execution cost, and tasks are assigned to processing elements with earliest expected completion time; this scheme is then combined with a frame-level priority assignment scheme which guarantees the execution of most relevant frames. They also introduce a simple but accurate task execution time prediction method, where the execution time of the current frame is determined by the execution time of a similar typed previous frame. However their simulations assume a bus-based communication architecture and constant rate DMA transfers and does not take into account contention patterns seen in platforms with Network-on-Chip based interconnects.

In [19, 21], observation based control algorithms have been used to improve the resource utilisation of the platform by providing statistical service guarantees to each client. These algorithms prove to be safe but in the case of the worst-case

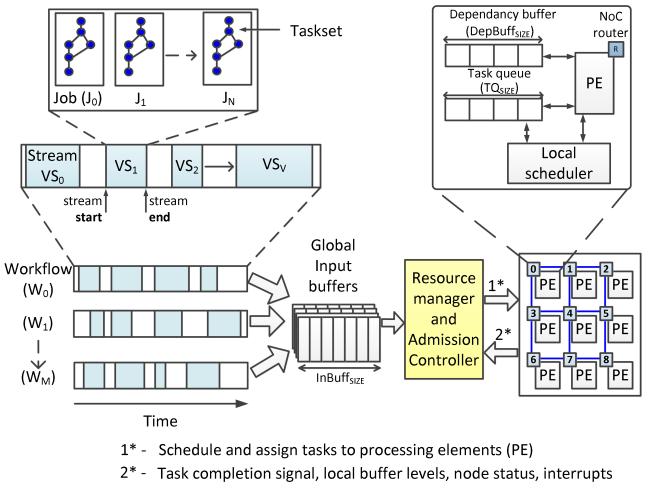

Figure 1: System overview diagram

scenarios it may cause deadline misses to occur. On the other hand, deterministic approaches such as in [5, 15], provides a guaranteed and predictable service by taking into account worst-case behaviour of the application and platform. If these worst-case assumptions of the workload are highly pessimistic, then the admission control algorithms will cause severe under-utilisation of system resources such as communication links and computation nodes.

Feedback control real-time scheduling has been implemented to continuously monitor the system characteristics and adjust the schedules and admission control decisions for unpredictable dynamic workloads [17]. The admission controller in these systems is aware of any unpredicted changes in the workload and is able to dynamically change the control decision in order to meet the required QoS levels. Feedback based control schemes have been used in multimedia based applications where the video encoding rate was dynamically adapted to suit the estimated network bandwidth, thus enabling the delivery of good visual quality across a congested network [20]. In [12] the system load is fed back into the MPEG decoder to decide which MPEG frames can be decoded in time or skipped if the deadlines cannot be met, thereby maintaining reasonable quality even during peak system load. The efficiency of the admission controller lies within the accuracy of the approximated analytical model of the real-time system [14], which is challenging because real-time systems are non-linear and time varying. Unlike the approaches described in this section our proposed heuristic based method does not depend on deterministic analyses nor does it depend on analytical models to make the admission control decisions.

3. SYSTEM OVERVIEW

3.1 Application Model

We consider a task model of M independent stream based work-flows, denoted W_i, each containing V number of video streams and each video stream containing N independent jobs, as shown in Figure 1. The system can process several video streams in parallel and this can be thought of as different decoding requests that arrive within the same time frame sent by different users of the system. We assume

the number of video streams and their start/end times are completely arbitrary, and therefore at a given time the system will be decoding multiple video streams simultaneously which are contained within the different parallel workflows. Hence the workflows act as a container for the delivery of video streams to the platform.

We assume data-parallel processing at the frame-level, where a task denoted as τ_i represents a single MPEG encoded frame that requires decoding. Each job, denoted J_i represents a group of dependant tasks (also known as an MPEG group of pictures - GoP). Hence each job contains a chain of tasks (see Figure 2) with certain dependency/precedence constraints such that a tasks' execution can only start *iff* its predecessor(s) have completed execution and their output data is available. In Figure 2 the circles (e.g. I_0, P_1 etc.) represent MPEG frame level decoding tasks and the arrows represent data dependencies - for example task I_0 will send its output data to task P_1 after completing its execution. As shown in Figure 2, certain tasks in the task-graph can be executed in parallel (e.g. P_4, B_2, B_3) if all the precedence constraints are met and there are sufficient available resources. Tasks are preemptive and have a fixed priority. We assume the priorities are unique and are predefined - as is the case when users or application developers of the system would assign the priorities when they are creating the tasks. Video streams do not have priorities, however the individual tasks contained within the video stream have a fixed pre-assigned priority. The same task priorities are used throughout all the GoPs in the video stream, such that the priority assignment between different GoPs of the same video stream does not change.

The following terms are used to define the attributed of tasks and jobs:

- p_i is the priority of τ_i
- t_i is the period of τ_i
- c_i is the worst-case execution cost of τ_i
- r_i is the response-time of τ_i
- a_i is the arrival time - when τ_i is dispatched to the input buffer (not necessarily, equal to the release time)
- l_i is the lateness of τ_i
- d_i is the relative deadline of τ_i
- D_{e2e} is the end-to-end deadline of J_i, calculated with respect to the frame-rate

The exact execution time of the tasks are unknown in advance as MPEG decoding times vary greatly based on spatial resolution, content and compression type etc. The number of workflows and jobs are not known in advance as video decoding requests will arrive randomly. We consider the video streams to be MPEG-2 encoded, where there are three kinds of frame types; namely I (Intra), P (Predictive) and B (Bi-directional) frames. The type of frame does not only influence the execution cost of the decoding task but also the precedence order. For example, as shown in Figure 2, dropping an I-frame causes severe image quality degradation as it results in the whole GoP being invalid. We also assume a closed, fixed GoP structure of *IPBBPBBPBBBB* (decoding order; hence GoP length = 12).

The spatial resolution of a video stream will correspond directly to the computation cost of the task and the payload

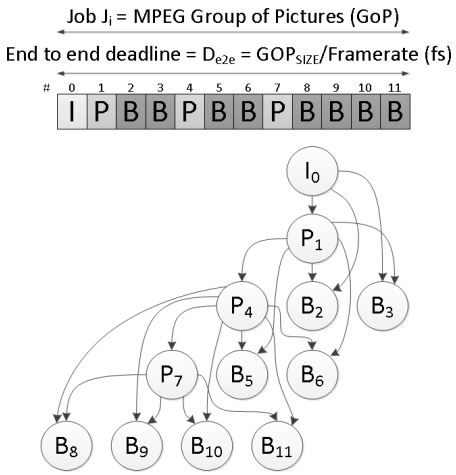

Figure 2: MPEG GoP data precedence graph

of the message flow, and this varies between different video streams. The execution time of the tasks within jobs across the same workflow and across other workflows is different, but has the same GoP structure, hence the same data precedence task graph (DPG) as shown in Figure 2. We assume there are no intermediate deadlines for each task in a job, however each job as a whole is considered schedulable if it completes execution on/before its D_{e2e}. We assume that all the tasks within a job will arrive at the same time instant but the arrival rate of a job will be sporadic, and the minimum inter-arrival rate is known in advance (analogous to using variable bit-rates with an upper bound).

Once a task has completed execution, its output (i.e. the decoded frame data) is sent as a message flow to the processing element executing its successor tasks τ_i^{secc}. If τ_i^{secc} are mapped on different processing elements, multiple messages will be released from the same task τ_i. We assume that each task produces a single or multiple messages which are sent immediately after it finishes its computation. Message flows inherit the priority of their source tasks, if multiple messages are produced by the same task then the message flows priority values are added an offset to ensure all flows have a unique priority.

A message flow denoted by Msg_i can be characterised by the following attributes:

- P_i is the priority of Msg_i
- T_i is the period of Msg_i
- PL_i is the payload of Msg_i
- C_i is the basic-latency of Msg_i
- R_i is the response-time of Msg_i
- D_i is the relative deadline of Msg_i

3.2 Platform Model

The distributed system is composed of P homogeneous processing elements (PEs) connected by a network on chip (NoC) interconnect. NoCs are the common communication architecture of choice for system on chips with dozens of cores [8]. The PEs are directly connected to the NoC switches which route data packets towards a destination core. NoCs offer a wide range of parameters to the system

designer such as topology, routing algorithms and switching strategies which permits flexible and scalable systems. We assume the NoC in our platform model uses fixed priority preemptive arbitration, a 2D mesh topology and uses the XY deterministic algorithm for routing. We assume that the NoC link arbiters can preempt packets when higher-priority packets request the output link they are using. The basic latency of a message flow, which is the time taken for a packet to be transferred from source to destination under the assumption of no contention over the NoC links is calculated according to Eq.1.

$$C_i = (numHops \times arbitrationCost) + numFlits \quad (1)$$

We assume each PE (e.g. CPU) is executing separate instances of the application software running on top of separate real-time kernels in an asymmetric multiprocessing (AMP) fashion. We assume the platform has no shared memory, and inter-processor communication (IPC) is performed via the NoC by passing messages. Each PE connected to the NoC contains a local memory, task queue and a dependency buffer. Once a task is released from a global input buffer, it is sent to the task queue of the PE, and the respective PE notified via an interrupt. The PE upon completing a tasks execution, transmits its output to the appropriate PEs dependency buffer.

The resource manager of the system, denoted as RM mainly performs scheduling, mapping and admission control of the video streams. The RM and admission controller (AC) is considered to be a single component in the platform, and the terms are used interchangeably throughout the paper. The RM is responsible for the timely release of tasks from the *global input buffers* to the PE task queues as well as mapping the tasks to specific processing elements. Tasks are held in the global input buffers until its predecessors have completed execution and their output data is transmitted to the required destination processing node. PEs will notifies the RM via interrupts once a task has completed its execution, so that the RM can release any successor tasks of the completed task. The RM performs a series of tests to decide if a new video stream request should be granted admission into the system. The task to PE mapping assignment is the same between GoPs of the same video stream and different mappings between GoPs of different video streams. This GoP level task to core mapping is established by the RM upon admission of a new video stream stream. A task-mapping table is maintained by the RM, and is queried by the PEs to determine where the completed tasks' output should be transmitted to. In this preliminary research work, we map each subsequent task to the node with the lowest number of tasks in its task queue. The goal behind this approach is to facilitate uniform utilisation across all nodes on the platform.

4. PROBLEM STATEMENT

As time progresses the multimedia system will receive multiple video stream decoding requests from different users of the system. The system should serve these video streams without missing any deadlines and dropping tasks. The first objective is that the AC decision should be predictable, such that if a video stream is admitted, the AC guarantees that the newly admitted stream will not incur any lateness throughout the streams lifetime, nor will the admission cause any lateness to the existing video streams already admitted. The second objective is to maintain high system utilisation of the processing elements of the system, which is desirable in an embedded multi-core system.

Improving *predictability* of the admission control algorithm in this research is defined as minimising the maximum and mean job lateness of the jobs that missed their deadlines, as well as minimising the number of dropped jobs under high load condition. Jobs will be dropped by the task-dispatcher at job release time if the global input buffers are full; tasks are not dropped once they have been admitted into the system. Dropped tasks negatively affects user-experience, predictability and system utilisation. A job is considered complete, when all its tasks have completed execution and a job is considered late if its end-to-end response time is greater than the end-to-end deadline. A completely predictable AC fully guarantees that all of the admitted video streams will be schedulable and would not incur any lateness or dropped jobs/tasks nor will it disrupt any of the existing streams. Furthermore a completely unpredictable AC does not guarantee the timely execution of the video decoding tasks admitted and will have a higher maximum job lateness, and more dropped jobs. The second objective - *system utilisation*, can be defined as the ratio between the number of active (busy) PEs and the total PEs in the system, in other words the secondary goal is to increase the system busy time.

5. ADMISSION CONTROL TESTS

5.1 Deterministic Admission Control Tests

The tasks and flows in our soft real-time systems model have fixed priority and preemptive scheduling, as described in Section 3.1. Each processing node in the system has a priority ordered task queue, and since fixed priorities are used higher priority tasks are guaranteed access to shared computation resources in the platform. Hence as shown in Eq.2, classical schedulability analysis [1] can be used to evaluate the worst-case response time r_i of the tasks. Here the set $hp(i)$ denotes the tasks that have a higher priority and in the same task queue as task τ_i. Active tasks in the system can belong to different video streams, hence a lower priority task of one stream can be blocked by a higher priority task of another stream.

$$r_i^{n+1} = c_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{r_i^n}{t_j} \right\rceil c_j \quad (2)$$

Similarly on the network on chip interconnect, contention occurs when several message flows try to access the same network resource at the same time. And due to the priority based preemptive arbitration of the routers, the NoC architecture is able to provide guaranteed throughput to the message flows of higher priority. Hence a message flow will have at most two interference sources - *direct* and *indirect* interference flows. Direct-interferers (denoted S_{id}) are higher priority traffic-flows that have at least one physical link in common with the observed traffic-flow. Indirect-interferers (denoted S_{ii}) are higher-priority flows that do not share any links with the observed traffic-flow but share at least one link with a traffic-flow in S_{id}. Shi et al. [16] introduced an analytical approach to derive an upper bound for the worst-case network latency of each traffic flow in wormhole switching, fixed priority preemptive NoC and shown in Eq.3. In this

equation J_i^R is the release-jitter, J_i^I is the interference-jitter and C_i is the basic latency of the message flow Msg_i as described in Eq.1.

$$R_i^{n+1} = C_i + \sum_{\forall j \in S_{id}} \left\lceil \frac{R_i^n + J_j^R + J_j^I}{T_j} \right\rceil C_j \quad (3)$$

Subsequently this analysis was extended in [10], where the response time r_i of the task τ_i that releases the message-flow is considered to be the release jitter of Msg_i, hence $J_i^R = r_i$, as shown in Eq.4. This is assuming the message flow is released immediately after the execution of its source task.

$$R_i^{n+1} = C_i + \sum_{\forall j \in S_{id}} \left\lceil \frac{R_i^n + r_j + J_j^I}{T_j} \right\rceil C_j \quad (4)$$

The worst-case end-to-end response time of a task comprises the worst-case computation time of τ_i and worst-case communication latency of Msg_i, as shown in Figure 3(a). Hence a the end-to-end schedulability of a task can be checked by $(R_i + r_i) \leq D_{e2e}$

5.1.1 Exclusion of Non-interferers

Figure 3(b) shows an example of the execution of MPEG frame decoding tasks I_0, P_1, B_2 and B_3 on 3 processing elements. I_0 and P_1 are mapped on PE_0 and PE_1 respectively, and both B_2 and B_3 are mapped on the same processing element - PE_2. This example follows the data precedence task graph shown in Figure 2. In this example since B_2 and B_3 are assumed to be mapped on to the same processing element, we can see that only a single flow is sent by I_0 and P_1 to the destination PE_2, where the B-frame tasks are mapped. However B_2 and B_3 cannot start execution until PE_2 receives the decoded frame data from P_1.

To simplify the task and flow interference analysis, we assume that there is no overlap between executions and invocations of different jobs within the same video stream, (Figure 3(c)). We assume that a new job of an live video stream will not be received and decoded by the platform while still currently executing the previous job invocation; although realistically the platform can decode multiple contiguous jobs of the same video stream in parallel. Due to this restriction in the analysis, when determining the higher priority interferers of tasks, the precedence constraints are taken into account such that dependant and successor tasks are excluded from the interference set *hp(i)*. An example of this is seen in Figure 3(b) where P_1, B_2 and B_3 will never interfere with task I_0. Similarly, for flows depending on the task precedence relationship some flows may not interfere with others. For example in Figure 3(b) the flow $I_0 \rightarrow P_1$ will not interfere with the flow $P_1 \rightarrow B_2, B_3$. These exclusions are taken into account when calculating the worst-case response time of tasks and flows, to make the analysis slightly tighter than in [16], but as the simulation results (Section 7) shows us, is still safe.

5.1.2 End-to-end Job Schedulability

Since we are only concerned with the end-to-end deadline D_{e2e}, the schedulability tests should check if the cost of the critical path of a job is less than or equal to the end-to-end deadline of the job. The critical path of a job is the path with the largest accumulated cost, where cost refers to the worst-case-response time of tasks and flows. For example in the task graph shown in Figure 2 the critical path of the job would start with the root node I_0 and end in any one of the leaf nodes (i.e. the B-frames)

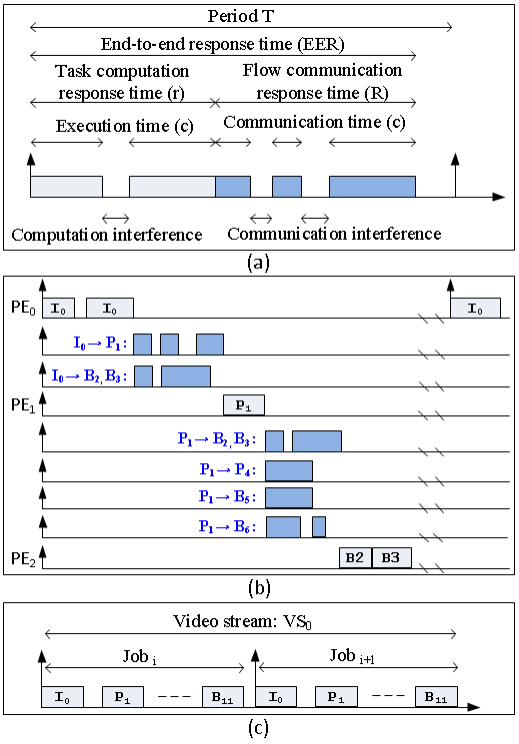

Figure 3: (a)End-to-end response time of a communicating task, adapted from [10] showing tasks and flows being interfered. (b)Example of 4 MPEG frame tasks executed over 3 processors (c)Non-overlapping job executions

The RM keeps track of each video stream that it has admitted and currently being processed. Each video stream is composed of a set of tasks and flows, and each task and flow can be blocked by tasks and flows from the same or other video streams. Therefore during the evaluation of the interference sets all tasks and flows of the system are taken into account. The RM performs the iterative calculations given in Eq.4 and Eq.2 when a new video stream admission is requested. However, this calculation takes a significant amount of time to compute when the interference sets of the tasks and flows are large in number. To avoid this issue a *timeout* is assigned to the calculation, where an intermediary result is returned when the timeout expires. The timeout needs to be set to a sufficiently high value such that a sufficiently accurate worst-case response time can be calculated within the time-limit but at the same time have a relatively fast computation time.

5.2 Heuristic Based Tests

The heuristic based tests described below are essentially best effort tests with minimum calculation overhead. They do not take into account the computational and communicational requirements of the tasks, but check the status of the platform to determine if it is overloaded or not, and makes the admission decision.

5.2.1 Instantaneous Task Lateness

A task is considered as late if it finishes its execution after its absolute deadline. However in our task model the jobs' end-to-end deadline is considered as an individual tasks relative deadline (i.e. $d_i = D_{e2e}$). This assumption causes tasks that are higher up in the task-graph to have longer slack times, and as a result the lateness calculation may be too optimistic. To alleviate this issue, we try to tighten the lateness calculation by multiplying d_i by a ratio as specified in Eq.5 and 6. l_i*(InputBuffer)* is the calculated instantaneous lateness of a task in the global input buffers, l_i*(TaskQueue)* is the calculated instantaneous lateness of a task in the task queues, t_c denotes the current time and a_i denotes the dispatch time of the task. IBL_α and TQL_α denotes the ratios used to calculate the relative deadline of the tasks in the global input buffers and task queues respectively.

$$l_i(InputBuffer) = (t_c - a_i) - (D_{e2e} * IBL_\alpha) \quad (5)$$

$$l_i(TaskQueue) = (t_c - a_i) - (D_{e2e} * TQL_\alpha) \quad (6)$$

Every task in global input buffers and the task queues incurs lateness as time progresses. Lateness values as calculated from Eq.5 and 6. Negative lateness values are considered as slack and are acceptable. The instantaneous lateness of all tasks within the input buffers and task queues are checked and if they are late, then this is taken as an indication of the system being overloaded. Subsequently, any new video streams are rejected until the system has sufficient free resources.

The problem of determining a suitable value for dependent subtask deadlines in distributed real-time systems has been addressed in the past by Kao and Garcia-Molina [13]. Their method, termed equal flexibility scheme (EQF) divides the total remaining slack among the subtasks in proportion to their estimated execution times. They use this deadline assignment scheme to reduce individual task deadline miss rate when the end-to-end deadline of a task-set is known, but the subtask deadlines are not known. Eq.7 shows the calculation of the absolute deadline using their EQF scheme; where we have substituted the worst-case-execution time c_i as the expected task execution time and m represents the total number of tasks in the task-set/job.

$$d_i = a_i + c_i + \left\{ \left[D_{e2e} - a_i - \sum_{j=1}^{m} c_j \right] \times \left[\frac{c_i}{\sum_{j=i}^{m} c_i} \right] \right\} \quad (7)$$

5.2.2 Free Space in Buffers and Task Queues

All tasks that arrive into the system are held in the global input buffers until they are ready to start execution. There exists certain scenarios, particularly during heavy workload conditions, where the global buffers do not have any space to accommodate new tasks (buffer overflow). If there is no available space in the global input buffer then the admission of new video streams should be avoided and if any jobs are due to be dispatched in that time instant, they will be dropped by the task dispatcher.

Once a task τ_i is ready to be executed (i.e. all the dependencies are available) it will be released to the respective PEs task queue. If the tasks priority p_i is higher than the other tasks in the task queue, then it will execute immediately, else it will wait in the queue with its status set to *blocked*. Once a task completes execution its output is sent to the dependency buffer of the processing element that is executing its child tasks. Similar to the task queues, the dependency buffers have a finite space. Hence during high load conditions the task queues of the processing elements and/or the dependency buffers may be full, and in these conditions any new video streams are rejected until the system has sufficient free resources.

5.2.3 Dropped Tasks

The system may become overloaded after a new video stream has been admitted. The admitted and live video stream may drop jobs if the global input buffers are full. It is important to note that tasks are not dropped after they have been dispatched into the system. From a user perspective, when jobs are dropped the video playback will freeze for the duration of the job/GoP (i.e. 0.48 seconds). The RM keeps track of any dropped tasks of a live stream. Dropped tasks of a live video stream is taken as an indication to an overloaded system, and subsequently the RM will reject any new video stream requests, until a given point in time where none of the live streams in the system have any dropped tasks.

All of the above heuristics are then combined and checked sequentially in the order they are presented such that if any one of them fails, a new video stream request is denied access.

6. EVALUATION METHOD

6.1 Abstract System-level Simulation

To explore the predictability and performance guarantees offered by the different proposed admission control tests, we implemented a high-level, discrete-event, abstract simulator. Figure 4 shows the interaction and execution sequence between the task dispatcher, resource manager, processing node and the NoC model of the simulator.

We implement the light-weight NoC simulation component as described in [11] to model the interference patterns of the traffic flows being transmitted on the NoC interconnect. This model considerably reduces the simulation time taken to model the NoC, as it only simulates the system at the time instants when packets enter or exit the NoC. The NoC model maintains the state of the NoC as a list of priority sorted flows, with additional information that is required by the algorithm such as their active status, interference sets, remaining payload etc. As time progresses the algorithm activates and deactivates the flows based on their interference set, and essentially simulates the activity of the direct and indirect interference of each flow via event firing. The NoC model notifies the RM of flows that have been completed, so that successor tasks can be released to the PEs, as shown in Figure 4.

The MPEG frame execution time calculation was selected randomly from a uniform distribution and the worst-case execution time (WCET) of a task in the taskset was chosen to be the highest randomly selected execution time. The bounds of the uniform distribution was set such that the WCET of an 720x576 I-frame was calculated as 0.097 seconds. The relationship between spatial resolution and com-

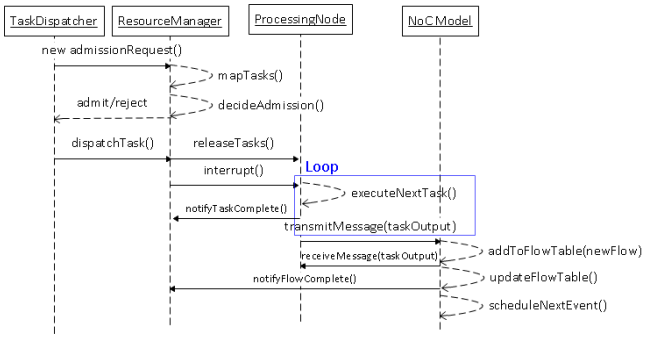

Figure 4: Execution sequence of abstract simulator

Parameter name	Value
Number of processing nodes	9 (3x3)
Task queue size	10
Dependency buffer size	10
Global input buffer size	12
NoC arbitration cost	7 clock cycles
NoC frequency	1000 MHz
Min/Max inter-GoP dispatch rate	(0.48, 0.72)
Min/Max inter-video dispatch rate	(0.53, 0.76)
Min/Max videos per workflow	(6, 7)
Min/Max GoPs per video stream	(7, 8)

Table 1: Fixed simulation parameters

putation cost was implemented at a MPEG-block level as described in [18].

The max/min videos per workflow and max/min GoPs per video stream given in Table 1 defines the number of simultaneous video streams active at any given time. As shown in Figure 1 video streams length and number of streams in a workflow can vary. The values given in Table 1 ensure that the video streams in each workflow will be densely packed to simulate heavy workload throughout the simulation time. The min/max video and GoP dispatch rates define the variable arrival rate of the videos and tasks respectively, and can be used to mimic the notion of variable bit rates and bandwidth fluctuations of the video streaming process. The maximum inter-GoP dispatch rate is used as the task and flow periods (i.e. t_i and T_i) for calculating the WCRT used in the deterministic admission control tests. Video stream resolutions were chosen randomly from different resolution combinations (e.g. 720x576, 426x240 and 320x240).

6.2 Experiment Design

The experimental work investigates the effect that the different two AC tests (heuristic and deterministic) had on several measurement metrics: predictable guaranteed video stream decoding service and system utilisation. Our definitions of predictability and system utilisation was previously stated in Section 4.

The heuristic-based admission control tests were evaluated for a range of IBL_α and TQL_α values and using the EQF subtask deadline assignment scheme given in Eq.7 which is be denoted as*Heu D-EQF*. The heuristic based tests are compared with the following baseline AC-tests: the situation where no admission control test was used (denoted *No-AC*) and when a deterministic AC test (denoted *Determ.*) is used. Hence the response variables in the experiment are predictability and system utilisation and the treatment is the different admission control test types. The hypothesis is that there exists a reasonable combination of the two heuristic parameters IBL_α and TQL_α, such that a sufficiently good compromise can be made between the two response variables. The load profile of the system is characterised by the number of simultaneous workflows that are active in the system at a given time. It is also important to note that the spatial resolution of the video streams processed by the system also directly affects the system load. The situation where 8 simultaneous workflows are present in the system is considered a *low load* condition since the number of workflows are less than the number of PEs in the platform; similarly 16 simultaneous workflows is considered a *high load* condition. We assume most often this soft real-time system will operate under low load conditions, however it is important to analyse and compare the system behaviour under high load conditions as well, where significant differences in deadline misses may occur. High load simulations consisted of 105 separate video streams each with an average of 84 tasks each. Each experiment was run for 35 different random unique seed values, and the results were later analysed by plotting the mean of all the runs.

The terms schedulable, late and rejected video streams are used throughout the results explanation. *Schedulable* video streams are those that were admitted and successfully processed without incurring any positive lateness or dropping any of their jobs. *Late* video streams are those which were admitted but some of the jobs had a positive lateness and/or jobs were dropped. *Rejected* video streams are those which the AC did not allow into the system due to either failing the admission control test or the system being overloaded.

Instantaneous system utilisation is difficult to measure in a system that deals with millisecond computation times, without causing the simulation to be slow and incurring large amount of data points. As an alternative, we measure the system busy time which is directly related to system utilisation. The busy time of each PE is tracked and added up at the end of the simulation time to analyse the total busy time of the system. A higher busy time means the system has a high utilisation, which is preferred as our secondary goal is to improve system utilisation.

7. EXPERIMENTAL RESULTS

The experimental results shown in Figures 5,6 and 7 need to be analysed in parallel to view the trade-off between predictability and utilisation. For example, we can see in AC-tests that show good performance in predictability show relatively low-utilisation levels.

Figure 5 shows a bar chart of the number of video streams that were successfully schedulable (green), late (blue) and rejected (red) for different of AC tests under light and heavy load conditions. The baselines, *No-AC* and *Deterministic* are shaded in light-grey. The ratios on the x-axis represents the IBL_α and TQL_α values respectively; for example *Heu(0.3, 0.7)* denotes the heuristic based admission control test results with $IBL_\alpha = 0.3$ and $TQL_\alpha = 0.7$. A full factorial parameter test was carried out (i.e 0.1 to 1.0 with step-size 0.1 for both ratios; all permutations), however due to space considerations only a selected few are shown. The data shown in Figures 5,6 and 7 represent the mean values of the 35 simulation runs.

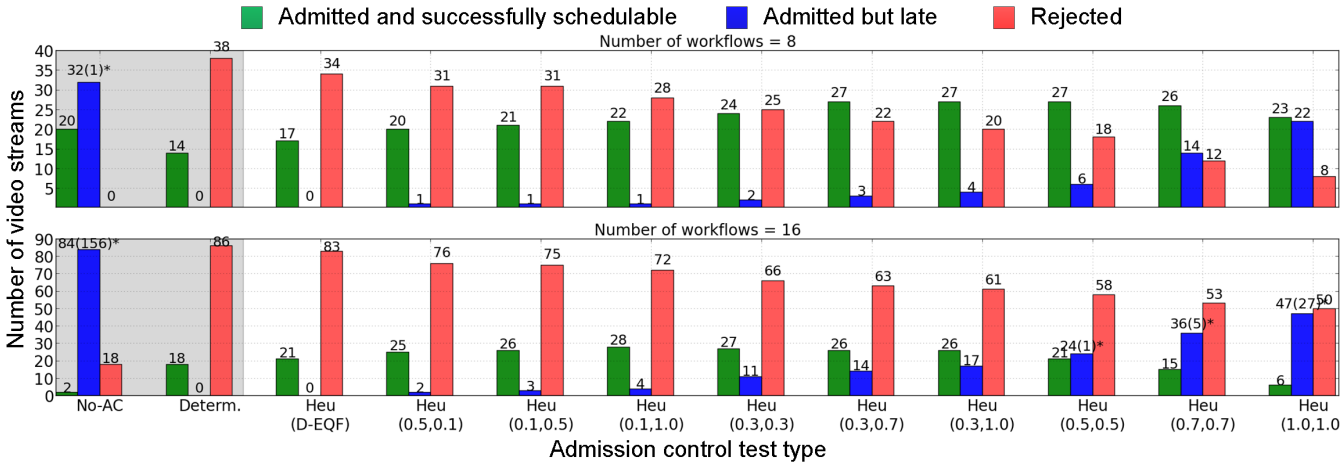

Figure 5: Predictability guarantee (video stream admission) results for different admission control tests. *value in brackets denotes number of jobs dropped due to global input buffers being full. Number of simultaneous workflows - 8 (low-load) and 16 (high-load)

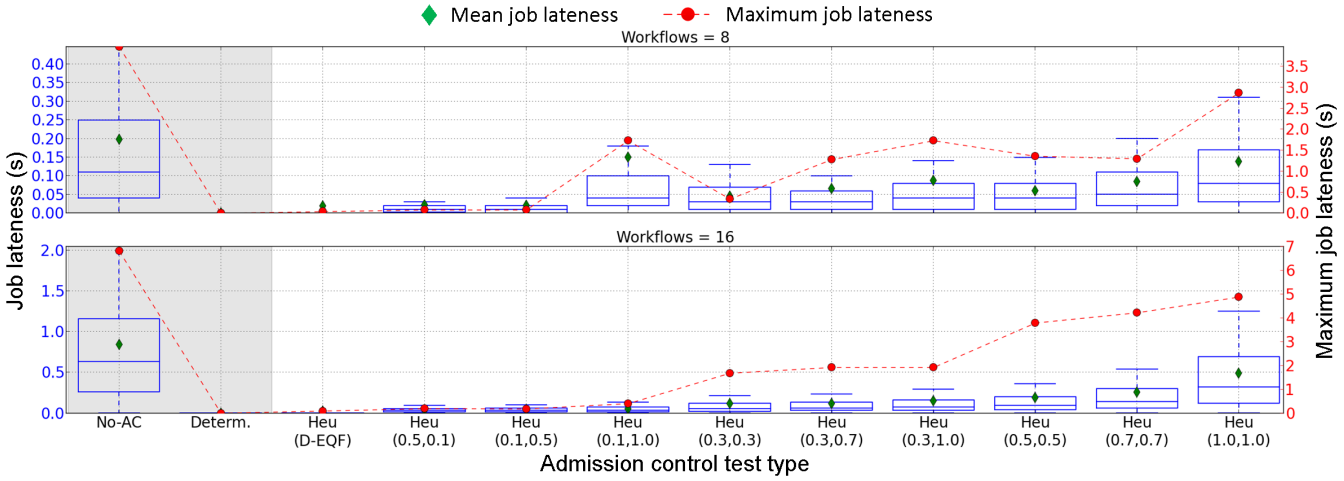

Figure 6: Completed job lateness distribution for different admission control tests. Number of simultaneous workflows : 8 (low-load) and 16 (high-load)

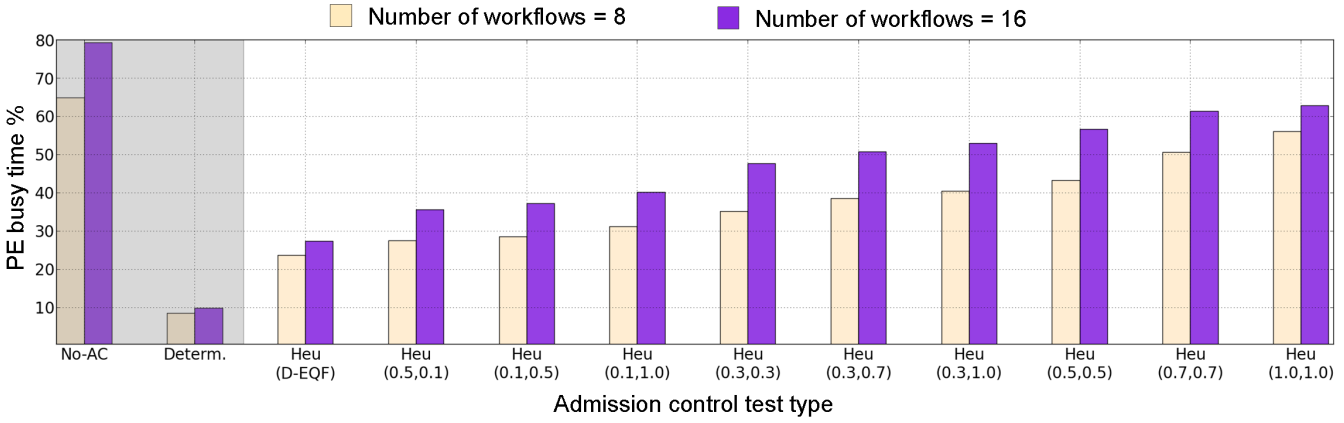

Figure 7: Percentage system busy time, for different admission control tests. Number of simultaneous workflows - 8 (low-load) and 16 (high-load)

In Figure 5 we can see that when the deterministic AC-test is used in both low and high load conditions - none of the admitted video streams incur any lateness or dropped tasks, but has a high rejection rate. Hence this test can be considered to be *safe*, and can be used to guarantee a high-degree of predictable service. When no admission control test is used (i.e. No-AC) the system admits all incoming video streams unless the global input buffers do not have available free space to hold the new tasks. Hence a large number of admitted video streams miss their deadlines under the high-load situation, while only few video streams are rejected. The No-AC test therefore provides no timing guarantee to the incoming video streams. As mentioned in Section 5.2.3, as video streams are admitted into the system, the load of the system will increase. When the task-queues and input buffers become saturated the active streams will start to drop new incoming jobs. For example, in the 16 workflow scenario when the No-AC test is used, there are on average, 156 jobs that were dropped by the admission controller due to the global input buffers being full, similarly in the Heu(1.0,1.0) AC-test we can see on average, 27 jobs being dropped. The results show that the Heu.D-EQF scheme offers similar service guarantees as the deterministic AC-test, where no admitted streams incur lateness, however on average, the rejection rate of this AC-test is lower than the deterministic AC-test, giving a more tighter guaranteed decision. The IBL_α and TQL_α ratio based AC-tests however showed a range of different results depending on the ratios used. The results shown in Figure 5 illustrates that smaller ratios lie closer to the deterministic tests while higher ratios show results similar to that of the No-AC test but with better (i.e. lower) rejection rates.

Figure 6 shows box-plots of the distribution of job-lateness for the different AC-tests under low and heavy load conditions. Lateness is calculated at a job level, and only jobs that incurred positive lateness are shown in this figure. The job lateness results shown in Figure 6 correlates well with the results shown in Figure 5. Also from Figure 6 we can see the maximum and mean job lateness increase in a similar manner to the distributions shown in the box plots. The maximum job lateness denotes the largest data point in the samples, and an increasing trend is seen as the IBL_α and TQL_α ratios increase. In the 16 workflow scenario, the streams that were admitted when No-AC test is used shows the most lateness while the deterministic AC-tests show no lateness to jobs as none of the streams that were admitted were late. The Heu.D-EQF AC-test shows similar results to that of the deterministic AC-test results, where none of the streams incurred lateness. We can see that the maximum lateness of the late jobs increase as higher values of IBL_α and TQL_α are used, such as in the Heu(0.5,0.5), Heu(0.7,0.7) and Heu(1.0,1.0) cases. However, because there are no late streams in the lower heuristic ratios, we cannot see any lateness data.

Figure 7 shows a bar chart of the percentage busy time of all PEs, compared between different AC-tests under low and heavy load conditions. The percentages shown is the ratio between the *total PE busy time* and the *total simulation time* displayed as a percentage. Higher values are more desirable as they correlate to higher utilisation. Admitting more streams into the system (regardless of whether they are late or schedulable), improves system utilisation. However if the admitted streams start to drop many jobs, as in the case of the No-AC and Heu(1.0,1.0) - 16 workflows case, then utilisation levels will drop due to the relative lack of activity in the system. For the tested workload the peak system utilisation is at 80% after which the buffers begin to overflow, as shown in the No-AC test case. Both the quantity of admitted video streams and their respective spatial-resolution (which corresponds to computation complexity), correlate directly to a more busy system. Admitting a few high-resolution video streams may cause the system to be more busy than when admitting relatively more lower-resolution video streams.

The deterministic AC-test has the lowest system busy time which corresponds well with the high number of stream rejections shown in Figure 5. When using the Heu.(D-EQF) AC test the system busy time is better than the deterministic AC-test which is because of the extra streams that were admitted into the system. In the high load condition the No-AC tests show the highest PE-busy time, as it rejects only 18 streams on average, due to buffer-overflow. Heuristic based AC-tests with higher values of the IBL_α and TQL_α ratios (e.g. *Heu(1.0,1.0)*, *Heu(0.7,0.7)*) show the highest amount of average system busy time consistently for high and low load conditions. Lower ratios on the other hand, show results similar to the deterministic AC-tests, since they too reject a relatively high number of video streams, and therefore the system is less busy.

7.1 Summary of results

The results in Figures 5, 6 and 7 confirm that deterministic admission control tests offer maximum predictable timing guarantees without any disruption to admitted streams. However this comes at a price of heavily under-utilising the system. On the other hand, using the No-AC test increases system utilisation significantly but at the cost of offering no service guarantees to the user. Heuristic based tests do not attempt to find optimality in both objectives but present a range of values which have different levels of predictability and utilisation. For example a combination such as $IBL_\alpha = 0.3$ and $IBL_\alpha = 0.7$ offers about 40% higher utilisation than the deterministic AC-test and 30% lower than the peak utilisation, with siginificantly less lateness increase (less than 0.45s on average).

The heuristic ratios IBL_α and TQL_α can be categorised into different ranges namely, *High*: $(1.0 \leq \alpha \leq 0.7)$, *Medium*: $(0.6 \leq \alpha \leq 0.4)$ and *Low*: $(0.5 \leq \alpha \leq 0.1)$. A general guideline that can be given when choosing appropriate values for these ratios is to select a higher value of IBL_α combined with med/high values of TQL_α; as this provides the best tradeoff between utilisation and predictability. On average the weight of the IBL_α ratio appears relatively more significant than TQL_α. A specific combination of these heuristics parameters can then be used to achieve a balance between high predictability and higher utilisation.

8. CONCLUSION AND FUTURE WORK

In this paper, we investigated a heuristic-based approach to compromise between predictability guarantees and system utilisation, when making admission control decisions on a NoC-based multi-core platform serving multiple dynamic video decoding requests. We have shown that our approach improves utilisation over the deterministic admission control test while maintaining acceptable lateness to real-time jobs processed, therefore showing better predictability guar-

antees than when no admission control is used.

The lateness calculation of the subtasks of the jobs that are live in the system depends on an accurate and optimum calculation of their deadlines. We use a ratio of the end-to-end deadline of the jobs to calculate the individual deadlines of the subtasks within the jobs, thereby enabling us to determine the instantaneous lateness of the tasks running in the system. We use this as a heuristic to make admission decisions. We have shown via system level simulation the performance comparisons between using a heuristic based admission control approach and the baseline deterministic and no admission control tests. We have also compared against a more fair slack distribution approach to calculating the subtask deadlines. Simulation results show that the heuristic based approach offers a range of utilisation and predictability values such that, the systems designer can choose the IBL_α and TQL_α thresholds depending on the requirements of the video streaming application and/or the load profile. For example higher ratios can be used if higher system utilisation is required or lower ratios can be used when predictable services are required. Potential future work in this research can be to explore better application specific, dynamic task mapping and priority assignment schemes to improve the latency and utilisation of the system.

9. ACKNOWLEDGEMENTS

We would like to thank the LSCITS program (EP/F501374/1) and DreamCloud project (611411), for funding this research and RheonMedia Ltd. for providing industrial case studies.

10. REFERENCES

[1] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Eng. Journal*, 8(5):284–292, Sept. 1993.

[2] C. Blanch, R. Baert, P. Coene, M. D'Hondt, Z. Ma, and R. Wuyts. Runtime scheduling for video decoding on heterogeneous architectures. In *Proc. of the 19th Int. Conf. on Real-Time Networks and Systems*, pages 195–204, 2011.

[3] S. Che, J. Li, J. Sheaffer, K. Skadron, and J. Lach. Accelerating compute-intensive applications with GPUs and FPGAs. In *Symp. on Application Specific Processors SASP 2008*, pages 101–107, June 2008.

[4] N.-M. Cheung, X. Fan, O. Au, and M.-C. Kung. Video coding on multicore graphics processors. *IEEE Signal Processing Magazine*, 27(2):79–89, Mar. 2010.

[5] J. Dengler, C. Bernhardt, and E. Biersack. Deterministic admission control strategies in video servers with variable bit rate streams. In B. Butscher, E. Moeller, and H. Pusch, editors, *Interactive Distributed Multimedia Systems and Services*, number 1045 in Lec. notes in Computer Science, pages 245–264. Springer Berlin Heidelberg, Jan. 1996.

[6] M. Ditze, P. Altenbernd, and C. Loeser. Improving resource utilization for MPEG decoding in embedded end-devices. In *Proc. of the 27th Australasian Conf. on Computer Science - Vol. 26*, ACSC '04, page 133–142, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[7] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler. Waiting times in quality of experience for web based services. In *4th Int. Workshop on Quality of Multimedia Experience, 2012*, page 86–96. IEEE, 2012.

[8] J. Henkel, W. Wolf, and S. Chakradhar. On-chip networks: a scalable, communication-centric embedded system design paradigm. In *Proc. of the 17th Int. Conf. on VLSI Design, 2004*, pages 845–851, 2004.

[9] O. Hohlfeld, E. Pujol, F. Ciucu, A. Feldmann, and P. Barford. BufferBloat - how relevant? a QoE perspective on buffer sizing. Technical report, Technische Universitat Berlin, 2012.

[10] L. S. Indrusiak. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *Journal of Sys. Arch.*, May 2014.

[11] L. S. Indrusiak and O. M. dos Santos. Fast and accurate transaction-level model of a wormhole network-on-chip with priority preemptive virtual channel arbitration. In *DATE 2011*, page 1–6. IEEE, 2011.

[12] D. Isovic and G. Fohler. Quality aware MPEG-2 stream adaptation in resource constrained systems. In *Proc. of 16th Euromicro Conf. on Real-Time Systems, ECRTS 2004*, pages 23–32, 2004.

[13] B. Kao and H. Garcia-Molina. Deadline assignment in a distributed soft real-time system. *IEEE Trans. on Parallel and Distributed Systems*, 8(12):1268–1274, Dec. 1997.

[14] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao. Feedback control real-time scheduling: Framework, modeling, and algorithms*. *Journal of Real-Time Systems*, 23(1-2):85–126, July 2002.

[15] D. Marinca, P. Minet, and L. George. Analysis of deadline assignment methods in distributed real-time systems. *Journal of Computer Comms.*, 27(15):1412–1423, Sept. 2004.

[16] Z. Shi, A. Burns, and L. S. Indrusiak. Schedulability analysis for real time on-chip communication with wormhole switching:. *Int. Journal of Embedded and Real-Time Comms. Systems*, 1(2):1–22, 2010.

[17] J. Stankovic, C. Lu, S. Son, and G. Tao. The case for feedback control real-time scheduling. In *Proc. of the 11th Euromicro Conf. on Real-Time Systems, 1999*, pages 11–20, 1999.

[18] Y. Tan, P. Malani, Q. Qiu, and Q. Wu. Workload prediction and dynamic voltage scaling for mpeg decoding. In *Proc. of the 2006 Asia and South Pacific Design Automation Conf.*, pages 911–916. IEEE Press, 2006.

[19] H. Vin, P. Goyal, and A. Goyal. A statistical admission control algorithm for multimedia servers. In *Proc. of the 2nd ACM Int. Conf. on Multimedia*, MULTIMEDIA '94, page 33–40. ACM, 1994.

[20] Q. Zhang, W. Zhu, and Y.-Q. Zhang. Resource allocation for multimedia streaming over the internet. *IEEE Trans. on Multimedia*, 3(3):339–355, Sept. 2001.

[21] R. Zimmermann and K. Fu. Comprehensive statistical admission control for streaming media servers. In *Proc. of the 11th Int. Conf. on Multimedia*, MULTIMEDIA '03, page 75–85. ACM, 2003.