Multi-Granularity Resource Reservations

Saowanee Saewong and Ragunathan (Raj) Rajkumar
Real-time and Multimedia Systems Laboratory
Carnegie Mellon University
{*ssaewong, raj+*}*@ece.cmu.edu*

Abstract

Resource reservation has been recently supported by many real-time operating systems to provide applications with guaranteed and timely access to system resources. Typically, reservations are based on the worst-case requirements, and therefore can inflate resource demands unnecessarily. Many multimedia applications such as MPEG video streams (1) have high worst-case to average-case demand ratio and (2) can tolerate some deadline misses. To support such applications, we propose a "multi-granularity" reservation model. Instead of the classical $\{C, T, D\}$ model of resource reservation, the multi-granular reserve specification is given by $\{\{C, T, D\}, \ldots, \{C^x, \epsilon^x T_i\}, \ldots, \{C^y, \epsilon^y T_i\}\}$ which represents a guarantee of the highest-granularity reserve for C units of resource during every successive periodic interval of T only as long as the resource usage by each of its low-granularity reserves (e.g., C^x units of resource in every recurring time of $\epsilon^x T_i$, $\epsilon^x \in Z^+$) is maintained. This multi-granular reservation approach delivers higher system utilization than the pessimistic strategy of worst-case reservation and better temporal isolation than other stochastic and heuristic guarantees in the literature. We perform a detailed schedulability analysis of this model using deadline-monotonic scheduling and derive an appropriate admission control test. We also present detailed analyses and simulation results comparing our reservation scheme for MPEG-4 streams with average-case resource reservation, constant bandwidth server (CBS), and (m,k)-firm guarantee.

1. Introduction

OS resource management for real-time and multimedia systems has been an active research area. Resource reservations based on worst-case requirements [15] have been widely supported by many RTOSs to provide timeliness guarantees and temporal isolation for hard real-time tasks. Unfortunately, this scheme over-estimates the desired resource requirements of soft real-time multimedia applications which generally have the following characteristics:

(a) highly varying resource consumption rate,

(b) large peak-to-average ratio of resource demand,

(c) long burst of large requests, and

(d) tolerance to occasional deadline misses.

For example, consider an MPEG video decoder with a worst-case resource reservation $\{C, T, D\}$ where C, T and D represent the maximum decoding time of a frame, the frame period and the frame deadline respectively. This reserve delivers timing guarantees for all frames, but this is obtained at a high resource cost. Consequently, it adversely affects the system utilization, hampers the progress of non-real-time tasks and limits the resources available to other reserves. This effect is unacceptable for interactive applications whose response times need to be low but execution patterns are difficult to predict to create a proper reserve.

A variety of techniques has been specifically developed to relax the QoS specification for multimedia streams to include their deadline miss tolerance and to diminish the overprovisioning of their resource needs. Those techniques can be categorized into two main approaches: deterministic lower-QoS specification and statistical approaches. Most deterministic lower-QoS specification approaches, such as the skip-over algorithm [13] and (m, k)-firm guarantee [11, 19, 21], allow users to specify the tolerance level of deadline misses and ensure the guarantees regardless of system workload. Their schedulability analysis pessimistically assume that all frames require the worst-case amount of resources. This results in resource overprovisioning per frame. The multiframe and generalized multiframe models [3, 16] allow users to provide more information about the variation of frame-to-frame execution time. While the former model requires the execution time of frames to be bounded in a fixed pattern, the latter requires a bound on the sum of the execution time of consecutive frames and therefore is more tolerant to tasks with dynamic execution patterns. Unfortunately, for some multimedia applications, determining such patterns of either the execution time or the sum of execution time on a frame-by-frame basis is not trivial. Moreover, differentiating the worst-case demand of *I, P* and *B* frames typically does not necessarily diminish the extent of overprovisioning due to the long-range-dependence found in MPEG4 streams.

Statistical approaches, such as average resource reservation and Constant Bandwidth Server (CBS), make reservation on resources based on the average demand. The delivery of high-demand frames therefore is not guaranteed and depends on instantaneous demand of other reservations and non-real-time tasks. Fundamentally, the schemes compromise the degree of temporal isolation offered in favor of high system utilization.

In this paper, we propose a "multi-granularity reservation" model. The scheme extends a dimension of the traditional resource reservation such that more detailed QoS requirements can be specified, analyzed and guaranteed deterministically. The proposed admission control allocates only the resource budget sufficient to satisfy the given QoS specification, efficiently manages resources across frames to handle fluctuating demands and still maintains the QoS isolation from other reserves and non-real-time tasks.

We now provide a brief overview of traditional resource reservation schemes and related work in the literature in order to provide context and background for the multi-granularity reservation design and analyses.

1.1. Resource Reservation Overview

Resource reservation used in a Resource Kernel (RK) [20] allows applications to independently specify their resource and timeliness demands, leaving the operating system to satisfy those demands among tasks using hidden resource management schemes. The reserve specification uses the $\{C, T, D\}$ model for the reservation of C units of resource time every recurring time interval T before a relative deadline D. To provide temporal isolation among tasks, the resource kernel monitors and enforces actual resources used by each reservation. When a task's reservation is depleted, the system will either halt or execute the task in background depending on its reservation type (hard or soft, respectively). Once its reserve budget is replenished at a new period boundary, the task resumes its reserved priority and reclaims its guarantee. In the rest of this paper, we use the term "reserve" and "reservation" interchangeably.

1.2. Related Work

Proposed techniques for multimedia guarantees can be categorized into two main approaches: deterministic lower QoS specification and statistical approaches.

The skip-over scheme [13], window-constrained scheduling [17] and (m, k)-firm guarantee [11, 19, 21] are examples of deterministic lower-QoS specification approaches. The skip-over algorithm specifies an application's tolerance using the skip factor s allowing one invocation skipped out of s. The (m, k)-firm guarantee represents an application that at least m out of any k consecutive invocations must meet their respective deadlines. These schemes classify requests as mandatory and optional invocations. Only the mandatory invocations are included in the schedulability analysis and obtain temporal guarantees. The optional invocations are executed in the background. Pessimistically, all mandatory requests are assumed to require the worst-case amount of resources and the number of guaranteed requests thus is fixed even when some requests actually consume less resources. On the contrary, our approach manages the resource budget efficiently on an inter-frame basis such that the left-over resources will be automatically transferred to succeeding requests as long as the budget of low-granular reserves is maintained and as a result more requests will be guaranteed.

Average-resource reservation and Constant Bandwidth Server [1] are examples of statistical approaches which allocate resources based on the average demand. An average-resource reservation basically is a soft reservation with a reservation equal to the average resource demand previously described in Subsection 1.1. Both schemes unfortunately can lead to unacceptably high service failure rate during a long burst of large requests in MPEG video streams.

Many statistical guarantees such as Distance Based Priority (DBP) assignment [10] and MPBP [12] schemes use best-effort heuristic methods. Some methods [2, 11] provide a probabilistic guarantee assuming the knowledge of probability distributions of the multimedia stream's inter-arrival and execution times. Unfortunately, obtaining such information in MPEG video streams [8] typically requires substantial effort and perhaps even impossible for live video applications. Due to this fact, our proposed scheme instead makes use of coarse-grained frame statistics such as the maximum frame size (or decoding time), the mean frame size and the size of the group of pictures (GOP).

Elastic scheduling proposed by Buttazzo, et al. [6] treats tasks as springs with different elastic coefficients. The model allows periodic tasks to intentionally change their execution rates to provide different QoS levels. This can be considered as a QoS negotiation technique which adjusts the QoS of existing tasks to accommodate a new task during overload. However, the model does not efficiently handle varying-resource requests and still rely on the uncontrollable task set characteristics at the negotiation time instant.

The well-known multiframe model proposed by Mok, et al. [3, 16] shares the same objective as ours which is to capture more information about the variation of frame-to-frame execution time. Their task model is given by $((C^0, C^1, \ldots, C^{N-1}), P)$ where C^i is the worst-case execution time of the i^{th} frame of the task. Its generalized model is relaxed to $< \Phi, P >$ where $\Phi = (\phi^1, \phi^2, \ldots)$ and ϕ^i lists the maximum total execution time of any i consecutive frames. While their model requires a fine-grained per-frame basis specification, ours allows a more coarse-grained approximation. This is especially useful for multimedia applications whose execution patterns are unknown in advance. One can consider our multi-granularity reserve as a special case of the generalized multiframe model. We provide a detailed schedulability analysis and derive both polynomial worst-case response-time test and utilization bound test. With a smaller domain of interest, our tests are similar but simpler than those proposed in [16]. In addition, we perform a detailed simulation analysis evaluating the performance of our scheme on real full MPEG4 video traces.

The rest of this paper is organized as follows. Section 2 presents an overview of our multi-granularity resource reservation. Section 3 presents a schedulability analysis of the model and derives an admission control test under the deadline-monotonic scheduling policy. Section 4 focuses on the deployment of multi-granularity reservation in multimedia applications such as an MPEG-4 video stream decoder. It also provides detailed analyses comparing our scheme with (m, k)-firm guarantees and CBS respectively. Section 5 shows the performance evaluation by simulation. Finally, we present our concluding remarks outlining our research contributions and future work in Section 6.

2. Multi-granularity Resource Reservation

This section gives an overview of the multi-granularity reservation concept including the design issues, the reserve model, and the mechanisms for reserve replenishment and enforcement. We now list some important considerations that influence the design of a multi-granularity reservation.

• **Flexibility of QoS Specification:** The classical Liu and Layland real-time task model [14] is generally intended for hard real-time systems where any deadline miss is undesirable. This model does not provide good support for QoS demand of multimedia applications whose timing constraints can be relaxed somewhat and therefore a more flexible QoS specification model is needed.

• **QoS Isolation and Deterministic Guarantee:** The ability to manage resources for maximum QoS return is a potential objective of real-time and QoS-guaranteed systems. To be able to collaboratively optimize deliverable QoS among applications with a middle-ware QoS manager, the OS must be deterministic such that the requested QoS is always delivered regardless of the behavior of other tasks.

• **Efficient Resource Utilization:** The main goal of real-time scheduling is to achieve high utilization while guaranteeing timing constraints and QoS requirements. Therefore, the system should satisfy the (pre-specified) resource requirements just enough to provide guarantees.

• **Varying Demand Tolerance:** Since multimedia applications can exhibit very fluctuating resource needs, an appropriate reservation scheme should modify resource allocations to suit these characteristics.

2.1. The Multi-granularity Reserve Model

Instead of using one reserve rate as in the classical reservation model, the multi-granular reserve specification is given by $\{\{C, T, D\}, \{C^x, \epsilon^x T\}, \ldots, \{C^y, \epsilon^y T\}\}$, where $\epsilon^x < \ldots < \epsilon^y$ and $\forall i, \epsilon^i \in Z^+$. The $\{C, T, D\}$ tuple is referred to as the highest granularity reserve and denotes an instantaneous resource demand of C units every recurring time interval T before a relative deadline D. The $\{C^x, \epsilon^x T\}$ tuple is referred to as a low-granular reserve and denotes the average resource demand of C^x units in the (longer) time interval of $\epsilon^x T$. With multiple granularities, the scheme gives flexibility to users to specify its average resource requirement(s) in the long run and simultaneously obtain guarantees for bursty requests as long as the average resource consumption over a longer time frame is maintained.

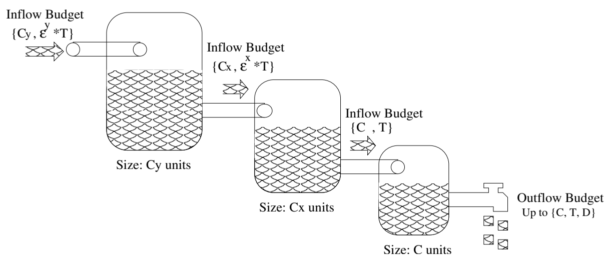


Figure 1. Cascading water tanks concept

Conceptually, the scheme behaves like cascading water tanks with different sizes. Figure 1 depicts this notion considering a reserve $\{\{C, T, D\}, \{C^x, \epsilon^x T\}, \{C^y, \epsilon^y T\}\}$. The resource in each tank is refilled repeatedly at its corresponding granular period interval. The amount of each refill depends on the availability of resource in its immediate preceding tank and its own leftover resource. This refill amount is limited to the available empty space in the tank at the time of replenishment. At the last tank, the outflow rate varies with user demand and also is limited to the rate of the highest-granular reserve $\{C, T, D\}$. For example, consider the x^{th}-granular tank, its tank size is C^x units and its refill rate is limited to C^x units every $\epsilon^x T$ time units. This tank is refilled only if there is some resource available at the y^{th} tank. If the user consumes less resource than C^x during a time interval $\epsilon^x T$, only that portion of C^x is needed to fill up the tank. Consequently, some of the resource in the y^{th} tank is automatically saved for future use.

With this cascading water tank concept, the multi-granularity reservation abstraction addresses all the designing goals described earlier. Users can flexibly control the budget in fine-grained fashion, enable precise resource allocation and achieve efficient system utilization. The tank sizes police resource consumption not to exceed the budget, establishing QoS isolation. The resource containment in the last tank supports the inter-period mutual budget enforcement mechanism and saves underused resources for the future, providing varying demand tolerance.

2.2. Replenishment and Enforcement

We now describe the implementation of the replenishment and enforcement algorithms that are integral to multi-granular reservation. For replenishment of a multi-granularity reserve with $k-$levels of granularity, k timers are utilized. Each timer has a period corresponding to its granular reserve period, $T, \ldots, \epsilon^k T$ respectively. At each granular timer interval (e.g. $\epsilon^j T$), a corresponding reserve budget (e.g. C^j) will be refilled. If its budget had been previously depleted, all higher granular reserves will be triggered to be resynchronized with its (j^{th}) replenishment timer. For enforcement, when a task associated with an undepleted multi-granularity reserve is executed, the amount of its granted resources is given by $MIN(c, \ldots, c^k)$ where c and c^j represent the existing budget of the highest- and j^{th}-granular reserve respectively. To keep track of reserve usage, at the end of each context switch, the used resource is deducted from the reserve budget of *all* granularities. If the task executes more than the granted amount, the task's priority is downgraded to the same level as (or below) that of other non-real-tasks. An example of the application of these rules will be provided in Section 3.2.

3. Schedulability Analysis

Our schedulability analysis follows an approach similar to that of Liu and Layland [14] and Mok and Chen [16]. However, the multi-granularity reservation model has some subtle assumptions and constraints. We assume the use of the deadline-monotonic scheduling policy and the existence of n multi-granular reserves, $R_1, R_2, \ldots, R_n$. Each reserve R_i is assumed to have g_i levels of granularity where $g_i \in Z^+$ (positive integer) and its reserve specification is given by $\{\{C_i, T_i, D_i\}, \ldots, \{C_i^x, \epsilon_i^x T_i\}, \ldots, \{C_i^{g_i}, \epsilon_i^{g_i} T_i\}\}$ where ϵ_i^x denotes the ratio of the x^{th} granular period with the highest granular one. By convention, we assume that $D_1 \leq D_2 \leq \ldots \leq D_n$. One task is associated with each reserve and denoted as $\tau_1, \tau_2, \ldots, \tau_n$ respectively. Note that a traditional reserve can be simply included in the analysis by considering it as a 1-level multi-granularity reserve. To obtain the analytical results, we assume that tasks are independent, periodic, and ready to run on arrival without blocking. In addition, we assume that they have the typical multimedia characteristics described in Section 1 and satisfy the *Accumulatively Monotonic* (AM) property. This property which is defined in [16] usually can be found in general MPEG streams. Intuitively, the AM property guarantees that the accumulative execution time of a sequence of requests in any time interval starting from its peak request is the largest among all other sequences with the same time interval. We also assume that the reserved *rate (C/T)* of lower-granular reserve is always smaller than that of a higher-granular one.

3.1. Critical Instant

There are two factors for a task associated with a multi-granularity reserve to miss its deadline: (1) the preemption of higher priority reserves and (2) its own insufficient budget. Assuming a task does not violate its resource quota constraints, one main goal of multi-granularity reserve is to provide the task its promised resource regardless of the behavior of other reserves and non-real-time tasks. We now determine the critical instant of such a task.[1]

Theorem 1 *Under the multi-granularity reservation model, assuming that all tasks associated with multi-granularity reserves are Accumulative Monotonic (AM), a critical instant for a task τ_i occurs whenever the task's request arrives simultaneously with the peak requests from all higher-priority tasks, their peak requests are synchronous with their lowest-granular periods and those tasks request resources at their maximum allowances.*

Proof.
We divide the proof into three parts. First, we prove that the peak requests of all higher-priority tasks must be synchronous with the target task's request to make their preemption be the longest. Secondly, we prove that the peak requests must also be synchronous with their lowest-granular period boundaries. Finally, without any knowledge of task execution pattern in advance, we prove that the higher-priority tasks must always have an accumulative demand at each period instant larger than their reserve budget.
Part I: Consider a higher-priority AM task, τ_j. Given the AM property, if the peak request of τ_j is ahead of τ_i's request, moving the peak request towards τ_i's request will make τ_i's preemption larger or the same. Likewise, if the peak request is behind, moving it towards τ_i's request cannot make the preemption shorter. Using this argument repeatedly across all higher-priority AM multi-granular tasks, the longest response time for τ_i occurs when it arrives together with the peak requests of its higher-priority tasks.
Part II: We can now assume that the peak requests of all higher-priority are synchronous with the target's request as shown in Part I. If the peak request of a high-priority task is

[1]The critical instant [14] is an instant at which a request from a task has its longest response time.

ahead of its lowest-granular period, moving the replenishment period towards it will cause the same or earlier preemption. On the other hand, if the lowest-granular period is ahead of the peak request, moving the lowest-granular period towards the peak request will give the high-priority task more budget and make τ_i's preemption larger.

Part III: With the nature of deadline-monotonic scheduling with resource enforcement, a high-priority task will always be able to preempt low-priority tasks as long as its reserve is undepleted. Therefore, packing all high-priority requests which follow their peak requests towards the peak requests can only make τ_i's response time longer or the same since succeeding requests can claim any leftover budget and get executed earlier. On the other hand, moving the succeeding requests away from their peak requests gives more chance for the target task to execute earlier and have a shorter or the same response time. Even without the knowledge of execution demand of tasks on a frame-by-frame basis, the largest preemption occurs when all higher-priority tasks request resource as much as they can, pack all requests at the beginning of τ_i's critical zone and have accumulative demand higher than their reserved budget. □

3.2. The Worst-case Response Time Test

In this section, we first demonstrate the worst-case response time test by an example. Consider a set of two tasks, τ_1 and τ_2, with reserves $\{\{3,5,5\},\{7,20\},\{13,50\}\}$ and $\{\{40,80,80\},\{60,160\}\}$ respectively. Figure 2 illustrates

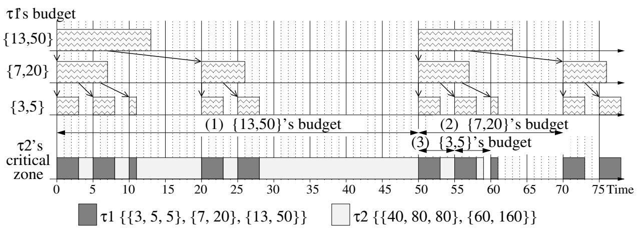

τ1's budget
{13,50}
{7,20}
{3,5}
(1) {13,50}'s budget
(2) {7,20}'s budget
(3) {3,5}'s budget
τ2's critical zone
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 Time
τ1 {{3, 5, 5}, {7, 20}, {13, 50}}
τ2 {{40, 80, 80}, {60, 160}}

Figure 2. The critical zone of task 2

the critical zone of τ_2's peak request as defined in Theorem 1. The preemption pattern by τ_1 during time interval $(0, t)$ is shaped in the ascending order of its multi-granular level of enforcement as shown in the bottom time line. The three time lines on the top show the available budget of τ_1 with cascading water tank replenishment. At each granular period, the budget is refilled from the immediate lower-granularity budget and the refill amount is limited to its granular budget quota. Consider the case when $t = 56$. Due to its lowest granular enforcement, τ_1's execution during time interval $(0, 56)$ cannot be larger than twice its lowest granular budget, which is given by $(\lfloor\frac{56}{50}\rfloor + 1)13 = 26$. While τ_1 will obtain the resource for 13 time units for the first interval of 50 time units, in the second interval its execution may be enforced by other higher granular enforcements. The next higher granular enforcement limits the execution in its second interval to $(\lfloor\frac{56-50}{20}\rfloor + 1)7 = 7$. However, the highest granular enforcement ensures that the task must not obtain resources more than 3 units each during time $(50, 55)$ and $(55, 60)$. In addition, during time interval $(55, 56)$, the maximum demand τ_1 can impose is 1 time unit. Consequently, the maximum time τ_1 can preempt τ_2 during time interval $(0, 56)$ is 17, which is given by $\lfloor\frac{56}{50}\rfloor 13 + \lfloor\frac{56-50}{20}\rfloor 7 + \lfloor\frac{56-50-0}{5}\rfloor 3 + MIN(5, 56-50-0-5)$. The first three terms basically represent the resource amount that τ_1 has been fully guaranteed by multi-granular periods, labeled as (1), (2) and (3) in Figure 2 respectively. The last term represents the maximum imposed demand for the partially guaranteed resources.

We now generalize the computation of preemption time encountered by a low priority task due to a high priority task τ_i during a time interval (t_0, t_0+t) assuming that both tasks' requests arrive at time t_0. Due to the cascading replenishment and enforcement nature of a multi-granularity reserve, τ_i's execution will always be enforced to the lower-granular budget during any elapsed time which contains full multiples of its corresponding lower-granular reservation period. Since this applies successively to all granular reservation tuples, the preemption will be shaped by multi-granular enforcements. $P^{(x)}$ denotes the preemption enforcement by the x^{th} granular reserve from a higher priority task τ_i, and g_i represents the lowest-granularity reserve of τ_i.

$$\begin{aligned}
P^{(g_i)} &\leq MIN(\lfloor\frac{t}{\epsilon_i^{g_i}T_i}\rfloor C_i^{g_i} + (t - m^{g_i}),\\
&\quad \lceil\frac{t}{\epsilon_i^{g_i}T_i}\rceil C_i^{g_i})\\
P^{(g_i-1)} &\leq MIN(\lfloor\frac{t}{\epsilon_i^{g_i}T_i}\rfloor C_i^{g_i} + \lfloor\frac{t-m^{g_i}}{\epsilon_i^{(g_i-1)}T_i}\rfloor C_i^{(g_i-1)}\\
&\quad +(t - m^{g_i} - m^{(g_i-1)}),\\
&\quad \lfloor\frac{t}{\epsilon_i^{g_i}T_i}\rfloor C_i^{g_i} + \lceil\frac{t-m^{g_i}}{\epsilon_i^{(g_i-1)}T_i}\rceil C_i^{(g_i-1)})\\
&\vdots\\
P^{(1)} &\leq MIN(\lfloor\frac{t}{\epsilon_i^{g_i}T_i}\rfloor C_i^{g_i} + \lfloor\frac{t-m^{g_i}}{\epsilon_i^{(g_i-1)}T_i}\rfloor C_i^{(g_i-1)}\\
&\quad +\ldots+\lfloor\frac{t-m^{g_i}-\ldots-m^2}{\epsilon_i^1T_i}\rfloor C_i^1 +\\
&\quad (t - m^{g_i}\ldots - m^1),\\
&\quad \lfloor\frac{t}{\epsilon_i^{g_i}T_i}\rfloor C_i^{g_i} + \lfloor\frac{t-m^{g_i}}{\epsilon_i^{(g_i-1)}T_i}\rfloor C_i^{(g_i-1)} + \ldots\\
&\quad +\lceil\frac{t-m^{g_i}-\ldots-m^2}{\epsilon_i^1T_i}\rceil C_i^1)
\end{aligned}$$

where $m^x = \lfloor\frac{t-m^{(x+1)}}{\epsilon_i^xT_i}\rfloor\epsilon_i^xT_i$, $m^{(g_i+1)} = 0$, $C_i^1 = C_i$ and $\epsilon_i^1 = 1$. With the collaborative enforcement across granularities, the preemption must satisfy all inequalities above. Note that the first term in the $MIN()$ operation from the highest granularity always has the smallest value compared to the first terms from others. Therefore, the maximum preemption by τ_i in time interval (t_0, t_0+t), denoted as $P_i^{(t_0,t_0+t)}$ is given by the following equation:

$$\begin{aligned} P_i^{(t_0,t_0+t)} \quad = \quad & MIN(\lfloor \frac{t}{\epsilon_i^{g_i} T_i} \rfloor C_i^{g_i} + \lfloor \frac{t - m^{g_i}}{\epsilon_i^{(g_i-1)} T_i} \rfloor C_i^{(g_i-1)} + \ldots + \\ & \lfloor \frac{t - (\sum_{j=2}^{g_i} m^j)}{\epsilon_i^1 T_i} \rfloor C_i^1 + (t - (\sum_{j=1}^{g_i} m^j)), \\ & \lfloor \frac{t}{\epsilon_i^{g_i} T_i} \rfloor C_i^{g_i} + \lfloor \frac{t - m^{g_i}}{\epsilon_i^{(g_i-1)} T_i} \rfloor C_i^{(g_i-1)} + \ldots + \\ & \lceil \frac{t - (\sum_{j=2}^{g_i} m^j)}{\epsilon_i^1 T_i} \rceil C_i^1, \ldots, \\ & \lfloor \frac{t}{\epsilon_i^{g_i} T_i} \rfloor C_i^{g_i} + \lceil \frac{t - m^{g_i}}{\epsilon_i^{(g_i-1)} T_i} \rceil C_i^{(g_i-1)}), \\ & \lceil \frac{t}{\epsilon_i^{g_i} T_i} \rceil C_i^{g_i}) \end{aligned} \quad (1)$$

Figure 3 shows the polynomial algorithm to determine the preemption time by a multi-granularity reserve. The worst-case response time of a task is the summation of its own computation time and possible preemption from all its higher-priority tasks. This leads to the proof of Theorem 2.

```
FindPreemption(t) {
To determine the worst-case preemption
time of a task, $\tau_i$.
/* p: preemption, k: granularity, enf:
    enforce */
  p = tmp = 0; enf = $\infty$
  For $k = g_i$ to 1
    tmp = tmp + $\lfloor \frac{t}{\epsilon_i^k T_i} \rfloor C_i^k$
    If (tmp $\leq$ enf) then
      tmp = enf; p = p + tmp; return p
    End if
    t = t-$\lfloor \frac{t}{\epsilon_i^k T_i} \rfloor \epsilon_i^k T_i$
    If (t==0) then p = p+tmp; return p End if
    If (k==1) then
      tmp = tmp + $C_i$
      If (tmp > enf) then tmp = enf End if
      p = p + tmp
      return p
    End if
    If (tmp + $C_i^k$ < enf) then
      p = p + tmp; tmp = 0; enf = $C_i^k$
    End if
  End for
}
```

Figure 3. Multi-granular Preemption

Theorem 2 *For a multi-granularity resource reservation system, the worst-case response time for a task τ_i is the smallest solution to the following equation:*

$$\omega^{(k+1)} = C_i + \sum_{j<i} P_j^{(0,\omega^{(k)})}$$

where $P_j^{(0,\omega^{(k)})}$ is the preemption time from higher priority task τ_j in the interval $(0, \omega^{(k)})$.

$P_j^{(0,\omega^{(k)})}$ can be determined by Equation (1) or the algorithm listed in Figure 3. The formula in Theorem 2 is solved recursively starting with $\omega^{(0)} = C_i$ and terminating when $\omega^{(k+1)} = \omega^{(k)}$ on success or when $\omega^{(k+1)} > D_i$ on failure. An appropriate blocking term must be added if synchronization needs introduce the potential for priority inversion.

3.3. The Utilization Bound Test

We now derive a simple utilization bound test for a multi-granular reserve when the deadline of a task is the same as its period (i.e. $D_i = T_i$). Consider two tasks in the system, τ_i and τ_j with τ_i having higher priority than τ_j. The preemption rate encountered by τ_j from τ_i varies with its period T_j and is given by $\gamma_{ij} = \frac{C_i^x}{\epsilon_i^x T_i}$ where $x = MAX(k \in Z^+ | (1 \leq k \leq g_i) \wedge (\epsilon_i^k T_i \leq T_j))$. γ_{ij} represents the *effective granularity* of τ_i with regards to τ_j. Therefore, considering its preemptions from all higher-priority tasks, the schedulability of a task, τ_j, can be determined using the classical Liu and Layland utilization bound as follows.

$$\sum_{i|T_i \leq T_j} \frac{C_i^{(\gamma_{ij})}}{\epsilon_i^{(\gamma_{ij})} T_i} \leq n(2^{\frac{1}{n}} - 1)$$

where n is the number of tasks in consideration. The utilization bound test of all tasks must be performed to determine the schedulability of the task set. Note that our model does not create a deferrable server-like effect of back-to-back execution due to the AM property assumption in multi-granular tasks which always have high demand of resource at the beginning of its low-granular periods. This enables the usage of the L&L bound.

4. Using Multi-Granular Reserves

Multi-granularity reservations are specifically designed for applications requiring soft real-time guarantees where requests can miss deadlines occasionally. For example, MPEG-4 video streams [9] are encoded into I, P and B-type frames. An I-frame is encoded as a single image with no reference to any frames and typically has a larger size than other frames. A P-frame is encoded relative to a past reference P or I-frame. A B-frame is encoded relative to a past reference frame, a future frame, or both frames which are the closest I or P frames. Each video sequence is composed of a series of Groups of Pictures (GOP). A GOP is an independently decodable unit that can be of any size as long as it begins with an I-frame. One effective and simple example of a multi-granularity reserve for an MPEG-4 video decoder is reserving its maximum decoding time in frame period granularity and its average decoding time per frame for each GOP period as given by

$$R = \{\{dt_{max}, T, T\}, \{dt_{avg} * gop, gop * T\}\} \quad (2)$$

where gop, T, dt_{max} and dt_{avg} denote the size of GOP, the frame period, the maximum and average decoding time respectively. Since a multi-granular reserve always delivers resources as long as the budget is available, the above reserve always guarantees the successful decoding of the I frame in every GOP and any GOP that requires the decoding time than $dt_{avg} * gop$ (or less) is also guaranteed. In practice, dt_{avg} may not represent the exact average but be sufficiently large enough to guarantee the desired frame processing deadline miss rate. Also, three or more granular reserves could be used for handling different subset sequences of the GOP. In other words, a more stringent or relaxed QoS specification and even probabilistic guarantees can be obtained. Less pessimism will result if a detailed profile of the video stream is available.

4.1. Multi-RSV vs. (m, k)-Guarantees

All (m, k)-firm guarantee schemes mark m out of k consecutive requests as mandatory requests. Only these requests are considered as preemption to low priority tasks assuming the worst-case demand. Some (m, k) schemes (e.g. [19]) subtly assign mandatory requests to reduce interference among tasks. However, these schemes need the knowledge of phasing among applications which is somewhat difficult in practice.

Our scheme does not assume such knowledge and enables budget to be shared across periods. Consider an MPEG-4 multi-granular reserve as given in Equation (2). If we assume that all requests require the maximum demand, at least $m_{mpeg4} = \frac{dt_{avg}*gop}{dt_{max}}$ frames can be guaranteed. In Section 5, we will compare our approach with (m, k)-firm guarantees using (m_{mpeg4}, gop) parameter and show that our scheme always performs better despite using an equal amount of allocated resource. Note that a Pfair schedule [4] of multi-granularity reserve can also be modeled by adding one more mid-granular reserve to equally distribute budget over its low-granular period.

4.2. Multi-RSV vs. Multiframe Model

A multi-granularity reserve can be considered as a special case of the generalized multiframe model. Any multi-granularity reserve can be conservatively transformed into a generalized multiframe task model. Fundamentally, the worst-case execution time patterns of consecutive frames will be conservatively assumed by our worst-case response-time test. For instance, a reserve given by $\{\{3, 5, 5\}, \{7, 25\}\}$ can be converted to $<(3, 6, 7, 7, 7), 5>$. Note that our model also allows tasks to have deadlines different from their periods. In addition, The scheme provides a simplistic method for users to specify required QoS in coarse-grained fashion. It transparently searches for its worst-case possible demand pattern and can deliver miss-tolerant service during fluctuating demands.

4.3. Multi-RSV vs. CBS

The Constant-Bandwidth Server uses earliest deadline first (EDF) scheduling as opposed to fixed-priority scheduling. Our scheme shares the same concept with CBS in the sense that it is tolerant of varying instantaneous demand yet in the long run the multimedia workload will be shaped into its average-case specification. Through the dynamic adjustment of requests' deadlines, CBS handles high-demand requests as a lower-priority class in EDF fashion starving all other non-real-time tasks. Moreover, its guarantees for the multimedia stream is not necessarily deterministic. In other words, a highly dynamic real-time workload can cause an unacceptably high failure rate. CBS with resource reclaiming [7] also has this property. On the contrary, in the same scenario, multi-granularity reservations will be able to detect this failure rate and reject the request for guarantees triggering renegotiation of the desired QoS. This predictability aspect is desirable and perhaps even crucial for real-time and QoS-guaranteed systems. Section 5 discusses the performance comparison of both schemes in detail.

5. Performance Evaluation

In this section, we evaluate the performance of multi-granularity reservation scheme, and compare them against average-case single-granularity reservations, (m, k)-firm guarantees and CBS. We will refer to these schemes as MULTI-RSV, AVG-RSV, MK-RSV and CBS respectively. Simulations are used to study QoS and temporal isolation, the response time of non-real-time tasks and the system utilization among those schemes. We chose four MPEG-4 video traces from [18], *Jurassic Park, Silence of the Lamb, News* and *Lectures Room Cam*, to represent four different kinds of movies. The trace lengths are 60, 60, 15 and 60 minutes respectively. All video streams are encoded using the GOP pattern given by $IBBPBBPBBPBB$ while data in the video streams typically occur as $IPBBPBPBBBPBB$. Table 1 summarizes the frame statistics. Unless explicitly stated, five real-time tasks are randomly generated to achieve the given total real-time utilization. Each task has a uniform probability of having a short $(1 - 10ms)$, medium $(10 - 100ms)$ or long $(100 - 1000ms)$ period. Period values are uniformly distributed within each range. Five non-real-time tasks are

Frame statistics	Jurassic	Lambs	News	Lecture
compression ratio (YUV:MP4)	9.92	13.22	10.52	36.27
frame rate	25	25	25	25
run time (ms)	3.6e+06	3.6e+06	9e+05	3.6e+06
mean frame size	3.80e+03	2.9e+03	3.6e+03	1e+03
var frame size	5.1e+06	5.2e+06	6.3e+06	8.3e+05
CoV of frame size	0.59	0.80	0.70	0.87
min frame size	72	158	123	344
max frame size	16745	22239	17055	7447
Avg. Utilization	0.103	0.0905	0.1025	0.064

Table 1. Video Frame statistics

randomly generated in the same fashion but without reservations. We estimate the video decoding time of each frame using the linear relationship with its corresponding frame size as suggested in [5]. Four metrics will be utilized:

`miss`: the ratio of the number of deadline-missing frames to the total number of frames.

`missI`: the ratio of the number of deadline-missing I-frames to the total number of frames (of all types).

`missD`: the ratio of the number of un-decodable frames (due to a deadline miss or the failure of its reference frames) to the total number of frames.

`dyn`: the ratio of dynamic errors (defined in [11] as the failure of a system to satisfy timing constraints of at least m frames out of any k consecutive frames) to the total number of possible k-consecutive frame sets.

5.1. Evaluation of a Single Stream

We first evaluate the performance of one single video stream competing with real-time and non-real-time workload. We believe that this scenario is most likely to happen in most handheld devices and personal computers. We create a multi-granularity reserve as given by Equation (2). The parameters m and k are determined as explained in Subsection 4.1. For CBS and average-case reserve, we allocate resources based on its average demand. For all schemes, in order to avoid queueing effects, we drop video requests immediately if they miss their deadlines.

Figure 4 shows the performance of one Jurassic movie at four different real-time and total system utilization scenarios: low workload (RT-U=0.4, U=0.5), low workload high background (RT-U=0.4, U=1.5), high workload (RT-U=0.65, U=0.75) and full system load (RT-U=0.65, U=1.5). RT-U and U denotes the real-time task utilization and the total system utilization, with both excluding the video stream utilization. For clearer view, we zoom in the `missI` ratio axes in Figure 4(c) and plot the `dyn` ratio axes shown in Figure 4(d) using a logarithmic scale.

As expected, AVG-RSV and MK-RSV perform poorly at high system utilization. At high workload and full system load, more than 35% of frames miss the deadline and more than 65% of frames are un-decodable under both schemes. MK-RSV delivers for 4% more I-frames due to the worst-case demand assumption on resource allocation and therefore achieves a somewhat better *MissD* ratio. MULTI-RSV and MK-RSV satisfy the (m, k) constraint as expected. Even though our MULTI-RSV allocates resources for approximately the same amount as MK-RSV, it always achieves lower failure rates and yields better system utilization $4-30\%$ and $10-57\%$ less for the `Miss` and `MissD` ratios respectively. Unlike other schemes, the performance of CBS does not suffer from the presence of non-real-time tasks. This is due to its preference of high-demand workload over non-real-time tasks. Note that MULTI-RSV has no missing I-frames at all, which results in a better `MissD` ratio than CBS.

We now repeat the same experiment with the other three video streams. Figure 5 shows the results. For clarity, we zoom into the `missI` and `dyn` ratio axes (0.1 and 0.25 respectively). Overall, *Lecture* obtains the better performance than others due to its much smaller variance in frame size. However, since most of its I-frames represent the group of the largest frame size and are likely to miss deadlines, any loss in the major reference frames results in bad `MissD` ratios of the *Lecture* stream. Again, the performance under CBS is independent from the existence of the non-real-time workload and the graphs with respect to low and high background workload are identical. Note that MULTI-RSV successfully processes all I frames again.

It must be noted that even though buffering relaxes the timing constraints and reduces the variation of resource demands, due to the long-range-dependence property of MPEG4 streams, a very large buffer is required to effectively smooth the burst. Unfortunately, this solution is impractical for handheld devices. Therefore, we recommend that reasonable buffering be combined with multi-granularity reserve to minimize delay while maintaining high system utilization.

5.2. Evaluation of Multiple Streams

We now simultaneously run three streams, *Jurassic, News* and *Lecture* with RT-U=0.35 and U=1.50; the parameters are chosen to fully utilize the system reserve capacity.

Figure 6 shows the performance of algorithms while multiplexing across multiple video streams. Comparing both experiments, multiplexing video streams yields better performance overall, especially the reduction of the `MissI` ratio in MK-RSV. This is because of two main reasons. First, there is typically a low probability that video streams that are statistically independent of each other will request resources at the peak rate simultaneously. Secondly, the worst-case resource allocation per frame used by MK-RSV more properly manages priorities for a complete I-frame execution. Again, *Lecture* obtains better performance than others due to its smaller fluctuation in demand. MULTI-

RSV yet again successfully delivers all *I*-frames and also achieves a low missD ratio. In addition, it maintains (m, k) constraints. In summary, this experiment shows that despite the benefit of the video multiplexing in a server where multiple streams can share resources, efficient deterministic guarantees are nevertheless crucial for protecting unpredictable high failure rate during overload. Note that we plot the missI and dyn ratio axes using a logarithmic scale.

5.3. System Response Time Comparison

In this section, we compare the effects of CBS and MULTI-RSV approaches on the system response time. A random number of short non-real-time requests, 0 to 3 requests, is generated every $40ms$ to compete for CPU resources along with one video stream *Jurassic*. The sizes of the requests are uniformly distributed between 10 and $20ms$. The average CPU utilization of the video stream is about 0.11. A set of real-time tasks with total utilization of 0.5 is generated in the same fashion as in the first experiment. We measure the normalized response time defined as $Normalized\_response\_time = \frac{completion\_time}{required\_decoding\_time}$. As can be seen in Table 2, even though both schemes deliver comparable performance (5% difference) on multimedia streams, CBS delays the non-real-time responses by approximately 17%. MULTI-RSRV also performs better on the missD and missI metrics and is worse by 2

Algorithm	miss	missI	missD	dyn	Response
CBS	0.0572	0.0116	0.143	0.0	33.120
MULTI-RSV	0.0772	0.0	0.108	0.0	28.323

Table 2. CBS vs. MULTI-RSV Comparison

5.4. Performance Predictability

Unlike CBS, multi-granularity reserve trades some system capacity against the provision of a (m, k)-firm guarantee. In this section, we confirm the necessity of predictability in scheduling policies. We experiment with multiple video streams using CBS. A real-time task set again is randomly generated as in previous experiments with the highest utilization allowed by CBS.[2] As can be seen in Table 3, the performance of video streams uncontrollably depends on competing real-time tasksets despite of the same utilization. The worst-case performance[3] may even have 27% of frames missing deadlines, 10% of dynamic errors and $4 - 64\%$ of decodable frames lost. Such unpredictability is generally unacceptable in many QoS-guaranteed systems.

[2]No non-real-time task is created since it does not affect CBS performance as previously discussed.

[3]This is defined as the highest average ratio among all performance ratios.

Performance Statistics	Miss	MissI	MissD	Dyn
Best performance				
Jurassic	0.0939	0.0089	0.1975	0.0013
Lecture	0.0210	0.0023	0.0479	0.0006
News	0.1001	0.0106	0.198	0.0017
Mean performance				
Jurassic	0.1190	0.0204	0.2880	0.0050
Lecture	0.0424	0.0119	0.1310	0.0032
News	0.1473	0.0297	0.3422	0.0183
Worst performance				
Jurassic	0.1688	0.0445	0.4413	0.0231
Lecture	0.0816	0.0306	0.2617	0.0136
News	0.2743	0.0676	0.6419	0.1019

Table 3. The Performance of CBS at high load

6. Concluding Remarks

Reservation-based real-time operating systems provide applications with guaranteed, timely and enforced access to resources. Classical reservation schemes are well-suited for hard real-time tasks whose worst-case demands are acceptable, but are not suitable for multimedia applications whose resource demands fluctuate widely and can also tolerate occasional deadline misses. In this paper, we proposed the "multi-granularity reservation", a new reservation paradigm which supports flexibility in QoS specification while still delivering core deterministic guarantees. The predictability of the guarantees can be conservatively achieved in the form of a firm guarantee provision or achieved in fine-grained fashion by the statistical profiling of multimedia streams. We have derived the schedulability analysis in two forms: worst-case response time analysis and a simple utilization bound test. In addition to offering predictability, the scheme outperforms deterministic, stochastic and heuristic approaches under fixed-priority scheduling policies such as average-case reservation as well as (m, k)-firm guarantees. In addition, comparing the fixed-priority version of multi-granularity reservations to the Constant Bandwidth Server (CBS) which uses EDF, the optimal dynamic scheduling policy, the performance of our scheme generally is comparable and even outperforms in case of MPEG4 video streams. Reservation-based real-time operating systems such as resource kernels need to be extended to support multi-granularity reserves and practical experience from their deployment needs to be obtained.

Acknowledgements: This work was supported in part by DARPA, the Office of Naval Research and the National Science Foundattion.

References

[1] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the IEEE Real Time Systems Symposium*, Madrid, Spain, December 1998.

[2] L. Abeni and G. Buttazzo. Qos guarantee using probabilistic deadlines. In *Proceedings of the IEEE Euromicro Conference on Real-Time*, York, England, June 1998.

[3] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *The International Journal of Time-Critical Computer Systems*, 17:5–22, 1999.

[4] S. Baruah, N. Cohen, C.G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.

[5] A. Bavier, B. Montz, and L. Peterson. Predicting MPEG execution times. In *SIGMETRICS/PERFORMANCE'98, International Conference on Measurement and Modeling of Computer Systems*, June 1998.

[6] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51, 2002.

[7] M. Caccamo, G. Buttazzo, and L. Sha. Handling execution overruns in hard real-time control systems. *IEEE Transactions on Computers*, 51, 2002.

[8] M. Dai and D. Loguinov. Analysis and modeling of a MPEG-4 and H.264 multi-layer video traffic. In *INFOCOM*, 2005.

[9] F. H.P. Fitzek and M. Reisslein. MPEG-4 and H.263 video traces for network performance evaluation (extended version). Technical Report TKN-00-06, Technical University of Berlin, Germany, October 2000.

[10] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers*, 44, 1995.

[11] M. Hamdaoui and P. Ramanathan. Evaluating dynamic failure probability for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers*, 46, 1997.

[12] K. H. Kim, J. Kim, and S. J. Hong. Best-effort scheduling (m,k)-firm real-time tasks based on the (m,k)-firm constraint meeting probability. In *the International Conference on Embedded Systems and Applications*, pages 240–248, June 2004.

[13] G. Koren and D. Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In *IEEE Real Time System Symposium*, Pisa, 1995.

[14] C. L. Liu and J. Layland. Scheduling alghorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.

[15] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *the IEEE International Conference on Multimedia Computing and Systems Computing System*, May 1994.

[16] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23, 1997.

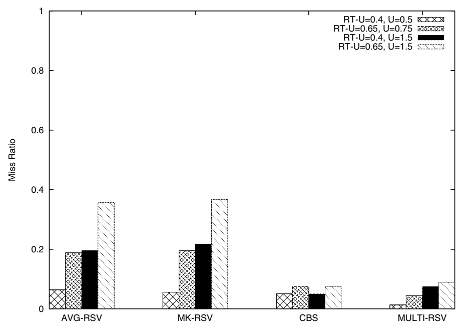
[17] A. K. Mok and W. Wang. Window-constrained real-time periodic task scheduling. In *IEEE Real Time System Symposium*, December 2001.

[18] Head of the Telecommunication Networks (TKN) Group. Mpeg-4 and h.263 video traces for network performance evaluation. http://www-tkn.ee.tu-berlin.de/research/trace/ack.html.

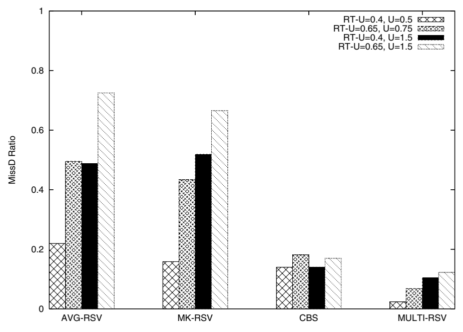
[19] G. Quan and X. (S.) Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *IEEE Real Time System Symposium*, November 2000.

[20] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, 1998.

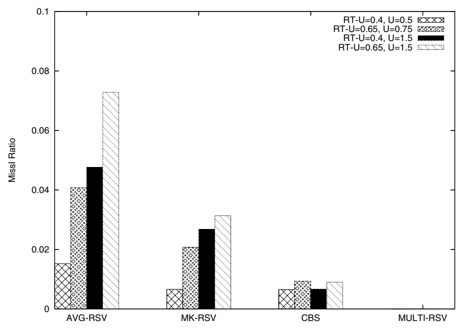
[21] P. Ramanathan. Overload management in real-time control applications using (m,k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10, 1999.



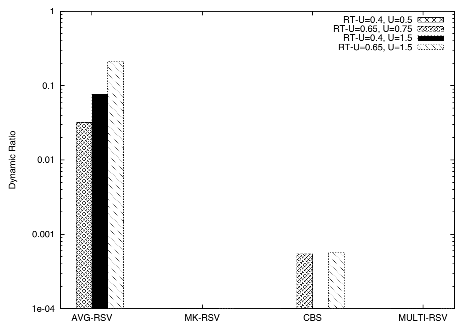
(a) Miss ratio



(b) MissD ratio

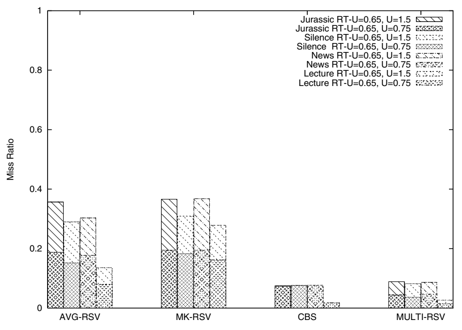


(c) MissI ratio

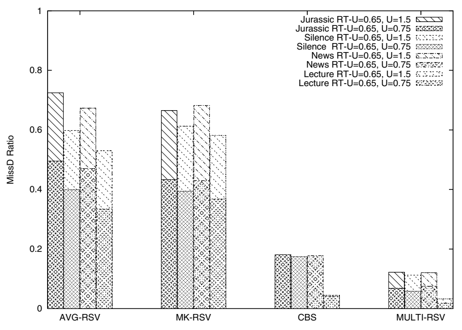


(d) Dynamic ratio

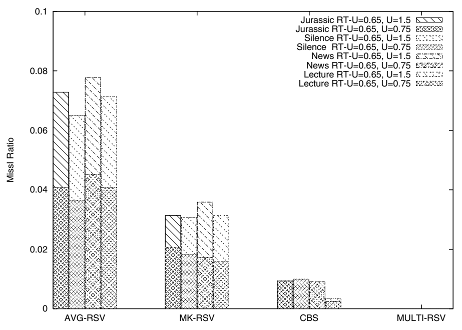
Figure 4. Jurassic's Miss ratios



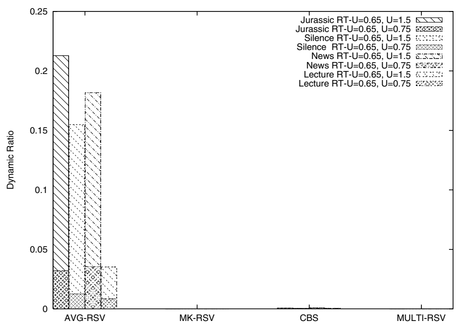
(a) Miss ratio



(b) MissD ratio

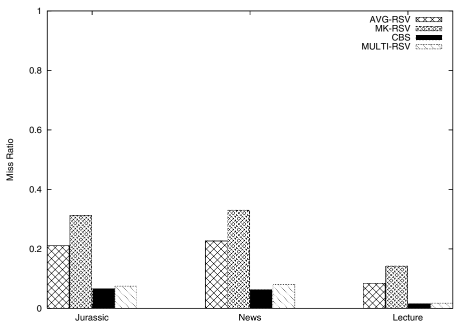


(c) MissI ratio

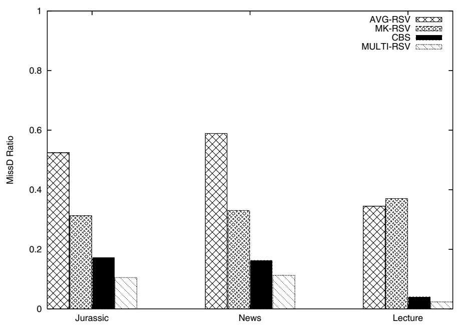


(d) Dynamic ratio

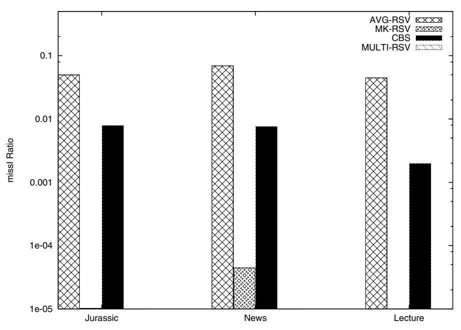
Figure 5. Video streams Comparison



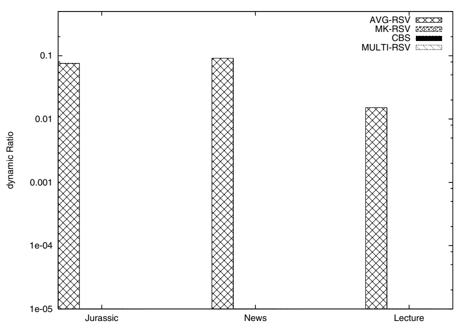
(a) Miss ratio



(b) MissD ratio



(c) MissI ratio



(d) Dynamic ratio

Figure 6. Multiplexing video streams