Cooperative Network and Energy Management for Reservation-based Wireless Real-Time Environments

Jun Yi, Christian Poellabauer, Xiaobo Sharon Hu, Dinesh Rajan,
Department of Computer Science and Engineering, University of Notre Dame
{jyi, cpoellab, shu, dpandiar}@nd.edu

Liqiang Zhang
Department of Computer and Information Sciences, Indiana University, South Bend
{liqzhang@iusb.edu}

Abstract

Reservation-based bandwidth allocation mechanisms in wireless and mobile environments, such as supported by the IEEE 802.11e standard, promise to offer enhanced support for real-time services and applications (e.g., mobile multimedia). This work is concerned with the scheduling of real-time traffic during the reserved medium access periods such that the applications' real-time communication needs are met. This is particularly challenging in systems where the bandwidth reservations are insufficient to meet all packets' deadlines. Further, this work observes that the increasingly popular energy management technique DVS (Dynamic Voltage Scaling) can further exacerbate this problem by delaying job executions and thereby packet generation (bringing them closer to their deadlines). Finally, wireless bandwidth is often affected by environmental inferences, which will further affect network performance. This paper studies these effects and presents an adaptive and cooperative mechanism to coordinate DVS, real-time packet scheduling, and link-layer adaptation, thereby increasing the number of packets meeting their deadlines, while ensuring that system-wide energy consumption is reduced.

1 Introduction

As the number of hand-held and mobile devices rapidly increases and wireless network hotspots are increasingly deployed, real-time media streaming applications on those devices will become more popular. It is challenging to support this and other real-time applications on wireless devices due to the unpredictability of the wireless medium. However, recent efforts have introduced resource (i.e., bandwidth) reservation mechanisms that can facilitate real-time streaming. For example, the proposed IEEE 802.11e standard [2] provides enhanced real-time and QoS support for real-time applications. This standard specifies a central control authority named the Hybrid Coordination Function (HCF) and offers contention-free medium access in the HCF Controlled Channel Access (HCCA) mechanism. In HCCA, the HCF (which typically exists at the access point) takes control of the channel and allocates transmission opportunities to each of the nodes in the network [2]. This is achieved by polling each node in a pre-determined order (e.g., round-robin) where each polling frame specifies the start and maximum duration of the channel access period, termed *Service Period (SP)*, allocated to a node. On reception of a polling frame, a node transmits its packets to HCF within the provided SP. At the end of a node's SP, the HCF polls the next node in its schedule and this process is continued for the remainder of the HCCA phase. The period of recurrence of the service periods at each node is referred to as the *Service Interval (SI)*.

While there have been numerous efforts on packet scheduling, including for real-time traffic, there is a dearth of research on packet scheduling in reservation-based systems. The challenge here is to allocate real-time packets to the available SP intervals such that all packets (in over-provisioned systems) or as many as possible (in under-provisioned systems) meet their deadlines.

This challenge is further exacerbated by the increasing use of energy management technique. Most notably, *Dynamic Voltage Scaling* (DVS) [4] has received wide attention and can be found in numerous wireless and mobile devices. However, as we will discuss below, the delay in job execution, and consequently in packet generation, can further complicate the real-time packet scheduling problem.

Finally, link adaptation [5] to dynamically vary the data transmission rate has been recognized as an effective way to improve the throughput performance of IEEE 802.11 and other wireless local-area networks (WLANs). There are a number of mechanisms to ensure proper adaptation of the transmission rate (e.g., adaptive rate selection among 11/5.5/2/1Mbps for 801.11b) in response to environmental interferences. The actual transmission time of packets may therefore vary and thus the effective allocated bandwidth of the device may vary as well.

In this paper, we consider a mobile device executing a set of *periodic* real-time tasks that generate real-time traffic in this 802.11e network. We assume that the device has already been allocated a pair of SP and SI values through a resource reservation mechanism by the access point. The goal is to transmit real-time packets in those SP intervals before their deadlines expire. Our proposed solution closely integrates existing DVS mechanisms on a wireless device with a novel packet scheduler and the wireless link layer. For example, decreasing the operating frequency of the CPU by the DVS algorithm will affect the timeliness of real-time packets. Increasing the operating frequency, on the other hand, leads to increased energy consumptions. Finally, the transmission rate and packet sizes (even of the same task) may vary and affect the effective bandwidth allocated to this device. This requires a packet management mechanism that coordinates task executions and packet transmissions to improve the timeliness of the packets and to maintain large system-wide energy savings.

2 Observations

In this section, we discuss our observations on the effects of real-time packet scheduling and the use of DVS. We use the following notations: $J_{i,j}$ represents the j^{th} job of the i^{th} task; $P_{i,j}$ represents the packet generated by job $J_{i,j}$; $AS_{i,j}$ and $WS_{i,j}$ represent the actual and worst-case size of packet $P_{i,j}$; $G_{i,j}$ represents the packet generation time of $P_{i,j}$ (i.e., the time a job submits a packet to the packet queue); $D_{i,j}$ represents the transmission deadline of packet $P_{i,j}$; and $d_{i,j}$ represents the deadline of job $J_{i,j}$. We further assume that a job can generate a packet at any time during job execution. DVS mechanisms have been used in

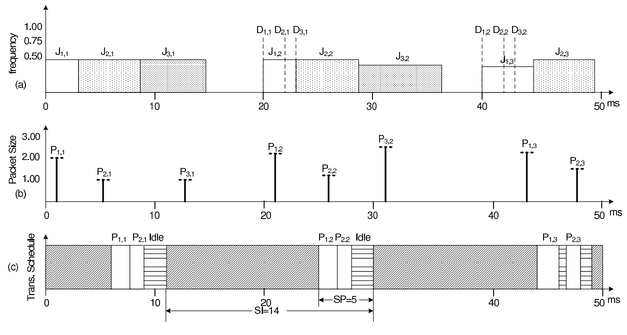

Figure 1. Effects of a DVS mechanism on real-time packet scheduling: (a) job schedule and clock frequency, (b) packet sizes (expressed as transmission times), (c) EDF scheduling for real-time packets.

the past to conserve energy at the processor level, including DVS approaches that ensure that the deadline requirements of real-time tasks are met [4]. Our observation, however, is that a DVS mechanism not only delays job execution, but also packet generation, thereby potentially causing some real-time packets to miss their transmission deadlines. This problem is exacerbated in reservation-based systems, where packet schedulers have only limited *transmission opportunities* during the SP intervals. That is, even slight delays in packet generation may push a packet out of its intended SP interval and prevent it from being transmitted before its deadline (if the next SP interval does not begin until the packet's deadline). As a consequence, packets will either be transmitted late or dropped altogether, e.g., Figure 1 illustrates a case where packets $P_{3,2}$ and $P_{3,3}$ miss their deadlines.

On the other hand, if we modify job deadlines such that packets can easily fit into their intended SP intervals (as illustrated in the example in Figure 2), the clock frequencies for job execution are increased (thereby increasing the energy consumption). Further, since the actual packet sizes may be less than their worst-case sizes, *idle intervals* within an SP interval may arise, i.e., durations where no transmission takes place, while energy at the wireless device is still consumed.

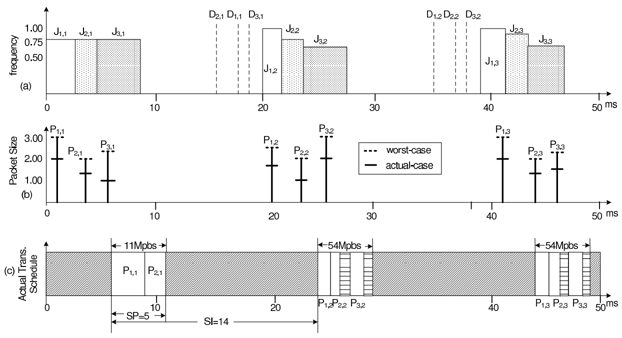

Figure 2. Effects of network scheduling mechanism: (a) job deadlines are modified to speed up job execution and packet generation, (b) actual and worst-case packet sizes (expressed as transmission times), (c) EDF scheduling of packets in the actual case.

Our final observation is that the network transmission rate also affects energy efficiency and real-time performance. Network transmission rate is in turn affected by environmental interferences [5]. With low transmission rates, it is better for the task management service to prolong the deadlines of jobs whose deadlines cannot be met (as illustrated in (c) of Figure 2), or to suspend some less critical real-time jobs. With high transmission rates, more real-time workload is allowed leading to less idle time in SP intervals.

3 System Model

Based on the our observations, we present an adaptive and cooperative model that integrates a processor-level DVS mechanism with a novel packet scheduler for reservation-based networks and the wireless link layer (WLL). When a new real-time job is entered into a run-queue (and it is expected that this job will generate a packet with real-time requirements), the DVS mechanism provides the corresponding packet parameters (Table 1) to the packet scheduler. Note that besides this new functionality, we do not make any further assumptions about the DVS algorithm. The *earliest ready time* is the worst-case packet generation time at the earliest possible job completion time (i.e., the packet is generated at the end of job execution at the earliest possible job completion). Note that the packet may be generated even earlier than its earliest ready time. One possible method to compute the earliest possible ready times is to run all jobs at the highest frequency using any desired task scheduling algorithm and then compute their completion times as earliest completion times. We further distinguish

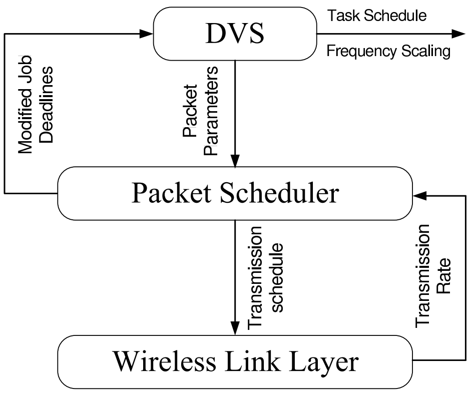

Figure 3. **Adaptive and Cooperative Model**

between two kinds of packets: packets which have not been generated yet are called Type-2 packets and already generated packets are called Type-1 packets. When a job is released (i.e., entered into a run-queue), it informs the packet scheduler of the Type-2 packet it will generate. When a packet is generated, it becomes a Type-1 packet and its actual size is recorded. The goal of the packet scheduler is to allocate all Type-1 and Type-2 packets into the available SP intervals and to provide this resulting transmission schedule to the WLL. Further, to ensure that all Type-2 packets will fit into their assigned SP intervals (remember that Type-2 packets have not been generated yet), the packet scheduler can inform DVS of a modified (i.e., earlier) job deadline, ensuring that DVS will run the job sufficiently fast such that the job's packet will be generated in time. DVS adjusts its frequency schedule according to this feedback information (where existing schedulability tests can ensure that the new frequency schedule will not violate any job deadlines).

Table 1. **Packet Parameters**

Name	Notation
Actual size	$AS_{i,j}$
Worst-case size	$WS_{i,j}$
Type	$TP_{i,j}$
Deadline	$D_{i,j}$
Weight	$W_{i,j}$
Earliest ready time	$E_{i,j}$

Finally, WLL uses a link adaptation mechanism to adapt the transmission rate to environmental interferences and feeds rate changes back to the packet scheduler. The packer scheduler, in response, can re-order packets in the transmission schedule. As a consequence, if the packet scheduler decides to push a Type-2 packet to a later SP interval, it may relax the corresponding job deadline again. Note that schedule changes triggered by WLL feedback cannot result in earlier job deadlines, i.e., only when a job is entered into a run-queue, the algorithm can set an earlier job deadline. This ensures that a job's deadline is not pushed earlier when a job is currently executing. Also, DVS can only move earliest ready times earlier to ensure all packets will be generated before their transmission times. Figure 3 outlines the proposed integrative DVS and packet scheduling mechanism.

4 Cooperative Energy and Real-Time Management

The primary goal of the integrative approach is to ensure that as many packets as possible can be transmitted during the limited transmission intervals. Each packet can have a weight associated, e.g., the weight could be the size of the packet or some user-specified urgency parameter. In this case, the goal is to increase the weighted sum of packets that meet their deadlines (note that finding an optimal solution to this problem is NP-hard, we therefore focus on heuristic solutions). The secondary goal is to increase the system-wide energy conservation, i.e., the combined energy saved at both the processor level and the network level. In *over-provisioned* systems (i.e, the SP periods offer more transmission opportunities than necessary to meet all deadlines), the main objective is to increase the energy savings as long as all deadlines are met. In *under-provisioned* systems, it may be impossible to meet all packets' deadlines and the objective is to increase the weighted sum of packets that meet their deadlines (while energy conservation is a secondary concern).

The packet scheduler's goal is to allocate packets to the available SP intervals and to adjust job deadlines. Packet allocation and deadline adjustment is triggered by these events:

- A new real-time job enters its run-queue, which results in a notification to the scheduler that a new Type-2 packet is available;
- a Type-2 packet becomes a Type-1 packet, where the actual packet size is less than its worst-case size, thereby opening up space in the SP interval for other packets;
- and the transmission rate is increased or decreased, which means that more or less packets can be transmitted and that the DVS mechanism may be allowed to adjust the frequency schedule.

For dynamic workloads of packets, there is no global deterministic and optimal packet scheduling algorithm when we assume that we have no knowledge about these packets (e.g., actual sizes and generation times). For a static workload, this problem can be reduced from the bin-packing problem and thus is NP-hard [1] (e.g., SPs are treated as bins and packets are treated as items of various sizes and weights).

Due to the large overheads [3] composed of medium access control (MAC) header, PHY preamble/header, acknowledgement (ACK) transmission, and some inter-frame spaces (IFSs), especially preamble and header are always transmitted at a much lower rate relative to the payload transmission rate. The size of packets will make a small difference on transmission times. As a result, we can reasonably assume that the transmission times of all packets vary in a narrow range, although their sizes vary in a wider range. If a packet of higher weight competes for the same time slot with a packet of smaller weight, allocating the slot to the packet of higher weight will almost surely result in higher total weight. To increase the weighted sum of packets being transmitted on time, packets of higher weight or with closer deadlines should have higher transmission priority. From this reasoning, we construct a heuristic algorithm for packet allocation and deadline adjustment, consisting of the following major steps:

- Step 1: Initialization. The transmission rate is updated. The release time $R_{i,j}$ of packet $P_{i,j}$ is its earliest ready time (i.e., $R_{i,j} = E_{i,j}$) if $TP_{i,j} = 2$ and otherwise the current time (i.e., $R_{i,j} = currTime$). The transmission duration $Tran_{i,j}$ of Type-1 packet $P_{i,j}$ is calculated based on actual size of its physical encapsulation, current transmission rate, and protocol overhead time. The transmission duration $Tran_{i,j}$ of Type-2 packet $P_{i,j}$ is calculated based on worst-case size instead.

- Step 2: Compute a weighted EDF-based schedule Tbl. The algorithm starts at the current time ($curPoint = currTime$) and scans all packets in increasing order of their deadlines. If the packet $P_{i,j}$ with the earliest deadline cannot meet its deadline (i.e., $curPoint + Tran_{i,j} > D_{i,j}$), the algorithm scans all packets ($PrePackets = \{P_{n,m} | [R_{n,m}, D_{n,m}] \cap Tbl[R_{i,j}, D_{i,j}] \neq \varnothing\}$) scheduled from the release time to the deadline of the packet, discards the packet ($ligtestPacket = argmin\{W_{i,j} | P_{i,j} \in PrePackets\}$) with the lowest weight, and then repack the schedule Tbl from the starting point (i.e., $startPoint(ligtestPacket, Tbl)$) of $ligtestPacket$, until $P_{i,j}$ can fit in or is discarded, whichever comes first. If $P_{i,j}$ fits in Tbl, the algorithm moves the current scheduling point further (i.e., $curPoint = curPoint + Tran_{i,j}$). This process repeats until the packet with the latest deadline is processed. This step's goal is to increase the weighted sum of packets that the SP intervals can accommodate.

- Step 3: Slack exploitation. This step scans all Type-2 packets $P_{i,j}$ in Tbl in decreasing order of their deadlines, and schedules each Type-2 packet as late as possible. If the successor $succ$ of a Type-2 packet in Tbl is a Type-1 packet, the algorithm first attempts to exchange the scheduling order of $succ$ and $P_{i,j}$ (i.e., $succ \leftrightarrows P_{i,j}$), as long as the deadline of $P_{i,j}$ is still met. If this fails, the algorithm attempts to move $P_{i,j}$ and its subsequent packets as late as possible ($P_{i,j} \rightarrow, succ \rightarrow, \cdots$), as long as the deadlines of those packets are all able to be met. This continues until the $P_{i,j}$ is unable to be moved later. If $succ$ is a Type-2 packet, the algorithm moves $P_{i,j}$ later, to at most the starting point of $succ$ or its own deadline (i.e., $\min\{D_{i,j}, startPoint(succ, Tbl)\}$). The above process repeats until the Type-2 packet with the earliest deadline is processed. The goal of this step is to delay required job deadlines as late as possible under the condition that the weighted sum of packets being transmitted on time are maintained.

- Step 4: Modify job deadlines and update the packet schedule. Here, the packet scheduler computes for each Type-2 packet a new job deadline, which is the beginning time of the SP interval a Type-2 packet occupies. The packet scheduler informs the DVS mechanism of these new deadlines. Further, the packet scheduler replaces the previous packet schedule with this resulting schedule Tbl and passes the schedule to WLL.

As a natural consequence of EDF, the resulting schedule is optimal if SPs are not overloaded.

The wireless link layer always selects the earliest packet from the current schedule as the next packet. When the next packet is of Type-2 and its earliest ready time is at least n milliseconds away (where n is a platform specific parameter), the network card can switch to a power saving modes if available.

5 Conclusions and Future Work

This paper investigates the conflicts between energy savings and real-time requirements of mobile devices in reservation-based wireless environments. We present our initial work on a collaborative approach to integrate processor-level energy management with network scheduling to ensure that as many packets as possible meet their deadlines, while energy consumption is kept low. The work in this paper can also be applied to generic bandwidth allocation situations in both wired or wireless environments. Our future work will extend the described approach by evaluating energy and real-time performance of our models and algorithms using experiments and simulations and investigating the effects of dynamically changing job deadlines on task scheduling and DVS.

References

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[2] IEEE 802.11 WG. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) enhancements for quality of service (QoS). *IEEE 802.11e Standard*, Nov 2005.

[3] Y. Kim, S. Choi, K. Jang, and H. Hwang. Throughput enhancement of IEEE 802.11 WLAN via frame aggregation. In *Vehicular Technology Conference*, pages 3030– 3034, New York, NY, USA, 2004.

[4] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP '01: Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, pages 89–102, New York, NY, USA, 2001. ACM.

[5] D. Qiao, S. Choi, and K. G. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *Transactions on Mobile Computing*, 1:278–292, Oct-Dec 2002.