Assessing the Impact of Dynamic Power Management on the Functionality and the Performance of Battery-Powered Appliances

A. Acquaviva, A. Aldini, M. Bernardo, A. Bogliolo, E. Bontà, E. Lattanzi
Università di Urbino "Carlo Bo"
Istituto di Scienze e Tecnologie dell'Informazione
Piazza della Repubblica 13, 61029 Urbino, Italy

Abstract

In this paper we provide an incremental methodology to assess the effect of the introduction of a dynamic power manager in a mobile embedded computing device. The methodology consists of two phases. In the first phase, we verify whether the introduction of the dynamic power manager alters the functionality of the system. We show that this can be accomplished by employing standard techniques based on equivalence checking for noninterference analysis. In the second phase, we quantify the effectiveness of the introduction of the dynamic power manager in terms of power consumption and overall system efficiency. This is carried out by enriching the functional model of the system with information about the performance aspects of the system, and by comparing the values of the power consumption and the overall system efficiency obtained from the solution of the performance model with and without dynamic power manager. To this purpose, first we employ a more abstract performance model based on the Markovian assumption, then we use a more realistic performance model – to be validated against the Markovian one – where general probability distributions are considered. The methodology is illustrated by means of its application to the study of a remote procedure call mechanism – through which a battery-powered device is used by some application requesting information – and of a streaming video service – which is accessed by a mobile client equipped with a power-manageable network interface card.

1. Introduction

Reducing the power consumption is one of the major criteria in the design of battery-powered devices typical of modern mobile embedded systems. This is achieved through the application of dynamic power management (DPM) techniques, i.e. techniques that – based on runtime conditions – modify the power consumption of the devices by changing their state or by scaling their voltage or frequency. The approaches to DPM proposed in the literature have been classified into deterministic schemes, predictive schemes, and stochastic optimum control schemes. The schemes of the first class schedule the shutdown periods at fixed time instants, possibly depending on the occurrence of some event. Instead, the schemes of the second class attempt to predict the device usage behavior in the future based on historical data patterns. Finally, the schemes of the third class make probabilistic assumptions – based on observations – about usage patterns to formulate an optimization problem. The interested reader is referred to e.g. [1] for a survey of the various DPM techniques.

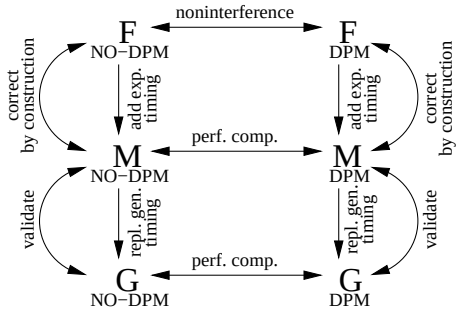

Figure 1. Incremental methodology.

Whenever a DPM technique is introduced in a battery-powered device, the functionality and the performance of the overall system may be altered. It is therefore important to be able to assess – in the early stage of the system design – the impact of the introduction of a DPM, in order to check that it does not significantly change the system functionality and that it does not cause an intolerable degradation of the system performance.

The objective of this paper is to provide an incremental methodology to study the effect of employing a DPM on both the functionality and the performance of a mobile battery-powered computing device. The proposed methodology is based on the application of formal analysis tech-

niques to a formal description of the system in which the considered device is embedded. Formal methods have been successfully applied to DPM systems in order to optimize DPM policies [9, 12]. In this paper, instead, we use formal methods to assess the impact of DPM. The application of the proposed methodology requires a sufficiently expressive specification language, in order to build models of the system functionality and performance, as well as a software tool equipped with the necessary analysis routines, in order to compare the properties of the models written in such a language. Although the methodology does not depend on a specific language, in order to illustrate it we use the Æmilia [4] architectural description language, together with its companion tool TwoTowers [2].

As shown in Fig. 1, the proposed methodology relies on the comparison of three different models of the system without and with DPM, each of which is incrementally obtained from the previous one by adding further details.

The first model of the system addresses only the functional behavior of the system. This model is used to check whether the introduction of the DPM alters the system functionality or not, regardless of any specific timing of the system activities. This is easily carried out by employing standard techniques borrowed from the field of noninterference analysis [8, 7]. More precisely, we verify whether the functional model of the system with the DPM activities being made unobservable is equivalent to the functional model of the system with the DPM activities being prevented from occurring (i.e., without the DPM). Should this not be the case, the modal logic formula generated by the equivalence checker to distinguish the two functional models can be used as a diagnostic piece of information to guide the modification of the DPM and/or the system in order to avoid mismatches. Establishing noninterference prior to the introduction of a specific timing, which may rule out some behaviors, ensures that the DPM is transparent in itself, not because of the adoption of ad hoc assumptions.

The functional model is then enriched by means of the specification of the timing of each system activity, which is provided through exponentially distributed random variables. The second model considered in the methodology is thus a Markovian model. This model does not need to be validated against the functional one, since it is directly obtained by attaching rates to the state transitions of the functional model. The Markovian model can be solved analytically (in an exact or approximate way) to compare the system without and with DPM on the basis of certain average performance measures – like energy consumption, system throughput, radio channel utilization, and quality of service – obtained when varying the DPM operation rates. Such performance indices can easily be expressed through a combined use of cumulative and instantaneous rewards.

The Markovian model is refined in turn by replacing – wherever necessary – the exponential delays with generally distributed delays, in order to more realistically reflect the actual delays characterizing the system and the DPM in practice. This general model must be validated against the Markovian one. This is carried out by verifying that both models result in comparable values for the considered average performance measures when substituting exponential distributions for general distributions in the general model. Once the validation succeeds, the general model (with realistic delays) can be simulated to estimate in a realistic setting the same average performance measures as before for different DPM operation rates, in the presence and in the absence of the DPM. The comparison of the resulting figures should guide the system designer in deciding whether it is worth introducing the DPM in a certain realistic scenario and, if so, in tuning the DPM operation rates without compromising the achievement of the desired level of quality of service.

The paper is organized as follows. In Sect. 2 we introduce the two case studies – a remote procedure call mechanism and a streaming video service – that will be used to illustrate the incremental methodology, together with the adopted specification language Æmilia. In Sect. 3 we apply noninterference analysis to both case studies, in order to check that the introduction of DPM does not change the system functionality perceived at the client side. In Sect. 4 we add exponentially distributed durations to the activities of the functional models for both case studies, thus yielding Markovian models to be used for comparing the values of the performance measures related to the power consumption and the overall system efficiency with and without DPM. In Sect. 5 we construct realistic models of both case studies by replacing the exponentially distributed delays with generally distributed delays. The general models are first validated against the corresponding Markovian ones, then they are used for comparing the estimates of the same performance measures as above with and without DPM. In Sect. 6 we draw some conclusions and we discuss the practical application of the proposed methodology.

2. Case Studies and Specification Language

The incremental methodology is illustrated through its application to two case studies, representative of real-world DPM systems: a power-manageable server receiving remote procedure calls from a blocking client and a mobile client accessing a streaming video server through a power-manageable wireless network interface card. In this section we introduce the two case studies, which will be hereafter denoted by *rpc* and *streaming*, respectively, together with the specification language Æmilia.

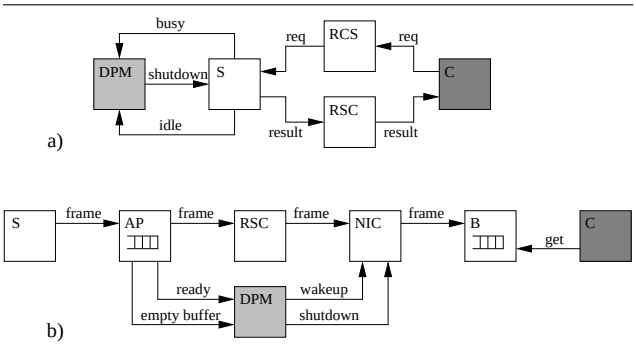

Figure 2. A picture of the two case studies: a) a power-manageable server for remote procedure calls; b) a streaming video server accessed by a mobile client through a power-manageable network interface card.

2.1. Remote Procedure Call (*rpc*)

Fig. 2.a) shows the topology of the *rpc* benchmark. The client (C) synchronously interacts with the server (S) through a full-duplex radio channel implemented by two half-duplex channels: RCS, from C to S, and RSC, from S to C. RCS is used by the client to send service requests (req) to the server, while RSC is used by the server to send the results (res) back to the client. The server also interacts with the DPM, which issues shutdown commands in order to put the server in a low-power inactive state. Two more signals (idle and busy) are used by the server to notify the DPM about every change of its state.

More precisely, the server has four main states: idle (in which it waits for service requests), busy (in which it delivers the requested service), sleeping (entered upon a shutdown command received from the DPM), and awaking (a transient state modeling a non-instantaneous wakeup from sleeping to busy, triggered by a service request arrival). When idle, the server is sensitive to shutdown commands. Depending on the application, the server may be also sensitive to shutdown commands when busy, in which case the shutdown interrupts the service.

On the other side, in its easiest implementation the blocking client issues a service request, waits for results, and takes some time to process the results before issuing the next request. A simple timeout mechanism can be introduced to resend a service request whenever the waiting time exceeds a given threshold. This can happen because the half-duplex radio channels are not ideal, hence they may introduce both a propagation delay and a packet loss probability.

The DPM sends shutdown commands according to a given policy, possibly based on the knowledge of the current state of the server. We shall consider two different policies. A trivial policy, where the DPM issues shutdown commands according to a given distribution, independently of the state of the server. A timeout policy, where shutdowns are issued upon expiration of a fixed (or randomized) timeout after the server has entered the idle state.

2.2. Streaming Video (*streaming*)

The top-level schematic of the *streaming* case study is shown in Figure 2.b). The server (S) asynchronously sends video frames (frame) to a buffer (B) that is local to the client (C). The video frames are transmitted through a wireless link modeled according to the IEEE802.11b [10] standard for wireless LANs. The server is directly connected to an access point (AP) that has an internal buffer possibly used to reshape the traffic, then AP uses a half-duplex radio channel (RSC) to send the video frames to the client network interface card (NIC) connected to B. The non-blocking client renders the video stream by taking (get) the video frames from B at a given rate.

According to the IEEE 802.11b standard, the NIC provides MAC-level DPM support that can be enabled via software. The actual implementation of the DPM support depends on the NIC. We model the protocol policy called PSP, which consists of placing the NIC in a low-power state – doze mode – in which it sleeps. Periodically, the NIC wakes up both to keep synchronized with the network and to check the AP buffer for possible frames received during the sleep period. In real systems, the support for the PSP DPM policy is implemented within the AP and the NIC. In order to abstract from implementation details and to carry out noninterference analysis, we model the DPM as an external component that takes information about the current state of the AP buffer and sends shutdown and wakeup commands to the NIC. A shutdown is issued as soon as the AP buffer becomes empty, while wakeups are sent periodically.

Since video frames must be reproduced at a given rate, frame requests (get) issued by the client represent real-time constraints that are violated whenever the local buffer B is not ready to provide the requested frame (frame miss). In addition, a frame may be lost because of a buffer-full event (frame loss).

2.3. The Specification Language Æmilia

We now provide a brief overview of the Æmilia [4] architectural description language, chosen together with its companion tool TwoTowers [2] to exemplify the incremental methodology proposed in this paper.

Since a thorough description of Æmilia is beyond the scope of this work (the interested reader is referred to [4]), we use the functional model of our *rpc* case study as an example to illustrate the key elements of the language. In par-

ticular, we consider a simplified version of the *rpc* system with ideal radio channels, a trivial DPM sending shutdown requests independently of the current state of the server, a server sensitive to shutdown commands both in the idle state and in the busy state, and a blocking client that does not use any timeout mechanism.

The Æmilia specification of the functional model, which starts with its name and the indication that there are no constant data parameters:

```
ARCHI_TYPE RPC_DPM_Untimed(void)
```

is composed of two sections. In the first section we define the behavior and the interactions of the types of components and connectors – generically called architectural element types (AETs) – that are in the system. The first AET that we define is related to the rpc server:

```
ARCHI_ELEM_TYPES
 ELEM_TYPE Server_Type(void)
  BEHAVIOR
   Idle_Server(void; void) =
    choice {
     <receive_rpc_packet, _> . Busy_Server(),
     <receive_shutdown, _> . Sleeping_Server()
    };
   Busy_Server(void; void) =
    choice {
     <prepare_result_packet, _> .
      Responding_Server(),
     <receive_shutdown, _> . Sleeping_Server()
    };
   Responding_Server(void; void) =
    choice {
     <send_result_packet, _> . Idle_Server(),
     <receive_shutdown, _> . Sleeping_Server()
    };
   Sleeping_Server(void; void) =
    <receive_rpc_packet, _> . Awaking_Server();
   Awaking_Server(void; void) =
    <awake, _> . Busy_Server()
  INPUT_INTERACTIONS
   UNI receive_rpc_packet; receive_shutdown
  OUTPUT_INTERACTIONS
   UNI send_result_packet
```

Then we define the behavior and the interactions for the ideal radio channel AET:

```
ELEM_TYPE Radio_Channel_Type(void)
 BEHAVIOR
  Radio_Channel(void; void) =
   <get_packet, _> . <propagate_packet, _> .
    <deliver_packet, _> . Radio_Channel()
 INPUT_INTERACTIONS
  UNI get_packet
 OUTPUT_INTERACTIONS
  UNI deliver_packet
```

Afterwards we have the definition of the behavior and the interactions of the blocking client AET:

```
ELEM_TYPE Sync_Client_Type(void)
 BEHAVIOR
  Sync_Client(void; void) =
   <send_rpc_packet, _> .
    <receive_result_packet, _> .
     <process_result_packet, _> . Sync_Client()
 INPUT_INTERACTIONS
  UNI receive_result_packet
 OUTPUT_INTERACTIONS
  UNI send_rpc_packet
```

Finally, we define the behavior and the interactions for the trivial DPM AET, which periodically shuts down the server even if this is busy:

```
ELEM_TYPE DPM_Type(void)
 BEHAVIOR
  DPM_Beh(void; void) =
   <send_shutdown, _> . DPM_Beh()
 INPUT_INTERACTIONS
  void
 OUTPUT_INTERACTIONS
  UNI send_shutdown
```

In the second section of the Æmilia specification we describe the system topology by declaring the instances of the previously defined AETs:

```
ARCHI_TOPOLOGY
 ARCHI_ELEM_INSTANCES
  S   : Server_Type();
  RCS : Radio_Channel_Type();
  RSC : Radio_Channel_Type();
  C   : Sync_Client_Type();
  DPM : DPM_Type()
```

as well as the attachments between their interactions:

```
 ARCHI_ATTACHMENTS
  FROM C.send_rpc_packet
   TO RCS.get_packet;
  FROM RCS.deliver_packet
   TO S.receive_rpc_packet;
  FROM S.send_result_packet
   TO RSC.get_packet;
  FROM RSC.deliver_packet
   TO C.receive_result_packet;
  FROM DPM.send_shutdown
   TO S.receive_shutdown
END
```

The topology is the same as shown in Fig. 2.a), up to the busy and idle triggers.

3. Comparing the Functional Models

The first model considered in the methodology of Fig. 1 describes the functional behavior of the system and is used to verify that the DPM is transparent, i.e. that its introduction does not have any influence on the system functional behavior as it is perceived by the client of the service provided by the system.

In order to assess the transparency of the DPM, we resort to the noninterference analysis approach [8] based on equivalence checking [7]. The general idea behind this approach is to view a system execution as an information flow and to consider that a group of system users (high users), using a certain set of commands, is noninterfering with another group of system users (low users), if what the first group does with those commands has no effect on what the second group of users can see. This approach has traditionally

been used for security purposes, as noninterference analysis can reveal direct and indirect information flows that violate the access policies based on assigning different access clearances to different user groups.

Within our methodology, the noninterference analysis is used to check whether the high system components, i.e. the DPM, interferes with the functional behavior of the system as observed by the low system components, i.e. the client. In our case studies, the actions of the DPM that modify the state of the power-manageable device are the only high actions, whereas all the actions of the client are the only low actions. Checking for noninterference amounts to verifying whether – from the client standpoint – the functional model of the system with the high actions being made unobservable (i.e. with the DPM being hidden to the client) is weakly bisimulation equivalent [11] to the functional model of the system with the high actions being made impossible (i.e. with the DPM being removed).

3.1. Application to *rpc*

The simplified version of *rpc* described in Sect. 2.3 fails the noninterference check. More precisely, when submitting the corresponding Æmilia specification to the security analyzer of TwoTowers, the outcome is negative and the following modal logic formula is returned:

```
EXISTS_WEAK_TRANS(
  LABEL(C.send_rpc_packet#
        RCS.get_packet);
  REACHED_STATE_SAT(
    NOT(EXISTS_WEAK_TRANS(
          LABEL(RSC.deliver_packet#
                C.receive_result_packet);
          REACHED_STATE_SAT(TRUE)
        )
    )
  )
)
```

The modal logic formula above means that the functional model of the simplified version of *rpc* with the high actions being hidden admits a computation path along which no result is returned to the client (synchronization of `RSC.deliver_packet` with `C.receive_result_packet`) after that the client has issued an rpc (synchronization of `C.send_rpc_packet` with `RCS.get_packet`), whereas this path does not exist in the functional model of the simplified version of *rpc* with the high actions being prevented from occurring. Recalled that the high actions are those performed by the DPM, the reason why the modal logic formula above distinguishes the two functional models is that in the latter model the DPM is not active, while in the former model the DPM is active and can shut the server down while it is processing an rpc. Since the client is blocking and does not use any timeout mechanism after sending an rpc, it may happen that it will be forever waiting for a response that will never arrive – only an rpc can wake the server up, but this cannot be issued by the client as long as it does not receive the response to its previous rpc.

Based on the modal logic formula and the considerations above, in order to make the DPM transparent to the client we first recognize that the client must implement a timeout mechanism – which by the way allows the client to cope with a more realistic radio channel that can lose packets – at the cost of complicating the server and the client, as they must now be able to discard old packets due to useless retransmissions. Second, we recognize that the DPM cannot shut down the server while it is busy, which is achieved by making the server inform the DPM about its state. As a consequence, we modify the AETs in the Æmilia specification of Sect. 2.3 as follows:

```
ELEM_TYPE Server_Type(void)
 BEHAVIOR
  Idle_Server(void; void) =
   choice {
    <receive_rpc_packet, _> . <notify_busy, _> .
     Busy_Server(),
    <receive_shutdown, _> . Sleeping_Server()
   };
  Busy_Server(void; void) =
   choice {
    <prepare_result_packet, _> .
     Responding_Server(),
    <receive_rpc_packet, _> .
     <ignore_rpc_packet, _> . Busy_Server()
   };
  Responding_Server(void; void) =
   choice {
    <send_result_packet, _> . <notify_idle, _> .
     Idle_Server(),
    <receive_rpc_packet, _> .
     <ignore_rpc_packet, _> .
      Responding_Server()
   };
  Sleeping_Server(void; void) =
   <receive_rpc_packet, _> . Awaking_Server();
  Awaking_Server(void; void) =
   choice {
    <awake, _> . Busy_Server(),
    <receive_rpc_packet, _> .
     <ignore_rpc_packet, _> . Awaking_Server()
   }
 INPUT_INTERACTIONS
  UNI receive_rpc_packet; receive_shutdown
 OUTPUT_INTERACTIONS
  UNI send_result_packet;
      notify_busy; notify_idle

ELEM_TYPE Radio_Channel_Type(void)
 BEHAVIOR
  Radio_Channel(void; void) =
   <get_packet, _> . <propagate_packet, _> .
    choice {
     <keep_packet, _> . <deliver_packet, _> .
      Radio_Channel(),
     <lose_packet, _> . Radio_Channel()
    }
 INPUT_INTERACTIONS
  UNI get_packet
 OUTPUT_INTERACTIONS
  UNI deliver_packet
```

```
ELEM_TYPE Sync_Client_Type(void)
 BEHAVIOR
  Requesting_Client(void; void) =
   choice {
    <send_rpc_packet, _> . Waiting_Client(),
    <receive_result_packet, _> .
     <ignore_result_packet, _> .
      Requesting_Client()
   };
  Waiting_Client(void; void) =
   choice {
    <receive_result_packet, _> .
     Processing_Client(),
    <expire_timeout, _> . Resending_Client()
   };
  Processing_Client(void; void) =
   choice {
    <process_result_packet, _> .
     Requesting_Client(),
    <receive_result_packet, _> .
     <ignore_result_packet, _> .
      Processing_Client()
   };
  Resending_Client(void; void) =
   choice {
    <send_rpc_packet, _> . Waiting_Client(),
    <receive_result_packet, _> .
     Processing_Client()
   }
 INPUT_INTERACTIONS
  UNI receive_result_packet
 OUTPUT_INTERACTIONS
  UNI send_rpc_packet

ELEM_TYPE DPM_Type(void)
 BEHAVIOR
  Enabled_DPM(void; void) =
   choice
   {
    <send_shutdown, _> . Disabled_DPM(),
    <receive_busy_notice, _> . Disabled_DPM()
   };
  Disabled_DPM(void; void) =
   <receive_idle_notice, _> . Enabled_DPM()
 INPUT_INTERACTIONS
  UNI receive_busy_notice; receive_idle_notice
 OUTPUT_INTERACTIONS
  UNI send_shutdown
```

and we introduce the following additional attachments:

```
FROM S.notify_busy TO DPM.receive_busy_notice;
FROM S.notify_idle TO DPM.receive_idle_notice;
```

thus faithfully reproducing the topology shown in Fig. 2.a).

We have verified through the security analyzer of TwoTowers that the revised Æmilia specification of the *rpc* case study meets noninterference, i.e. the DPM does not interfere with the functional behavior of the system as perceived by the client.

3.2. Application to *streaming*

Due to lack of space, we do not show the Æmilia specification of the functional model for the *streaming* case study, which satisfies noninteference and can be retrieved from `www.sti.uniurb.it/bernardo/twotowers/`

4. Comparing the Markovian Models

The second model considered in the methodology of Fig. 1 is a Markovian model obtained from the functional one by attaching exponentially distributed durations to its actions. In addition to that, further exponentially timed actions resulting in self-loops may be introduced in order to monitor the residence in certain states.

The Markovian model is thus consistent by construction with the functional one, in the sense that their underlying state spaces are isomorphic up to the possible additional self-loops and the action rates (which occur only in the labels of the transitions underlying the Markovian model). As a consequence, whenever the functional model meets noninterference, then so does the Markovian model.

In the specific case of Æmilia, it is also possible to express infinite rates – equipped with priority levels and weights – which are useful to model activities whose timing is negligible. Since the resulting immediate actions take precedence over exponentially timed actions, the use of immediate rates may alter the state space of the Markovian model with respect to the state space of the functional model. Therefore, whenever immediate actions come into play, the noninterference analysis must be repeated on the Markovian model.

The Markovian models with and without DPM can be solved through standard numerical techniques. The average performance measures of interest, which are related to the power consumption and the overall system efficiency, can easily be expressed by means of reward structures. In the specific case of Æmilia, which is based on stochastic process algebra, the reward-based measures are expressed by means of an auxiliary specification language inspired by [3]. By combining the solution of the Markovian models with and without DPM together with the reward structures defining the performance measures of interest, we can finally assess the impact of the introduction of the DPM on the system efficiency, aiming at finding a tradeoff between the power consumption and the quality of the delivered service.

Due to lack of space, in the following we leave out the Æmilia specifications of the Markovian models of the two case studies. Such specifications can be found at `www.sti.uniurb.it/bernardo/twotowers/`

4.1. Application to *rpc*

In order to evaluate the performance measure of interest for *rpc*, first of all we have to provide some values for the parameters of its Markovian models. Here we assume that the average server service time is 0.2 msec, the average server awaking time is 3 msec, the average packet propagation time is 0.8 msec, the packet loss probability is 0.02, the av-

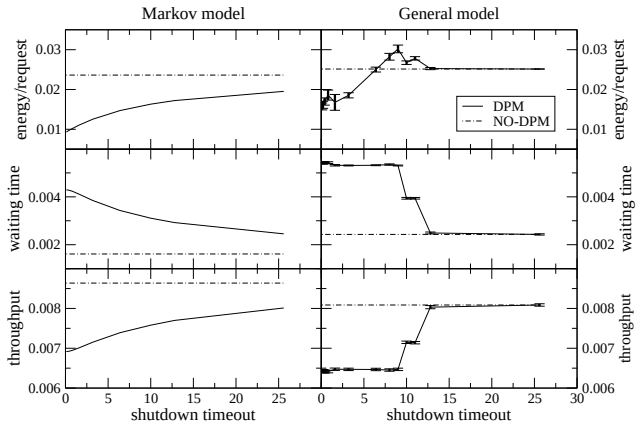

Figure 3. Performance comparison of the *rpc* benchmark with and without DPM for the Markovian model (left) and the general model (right).

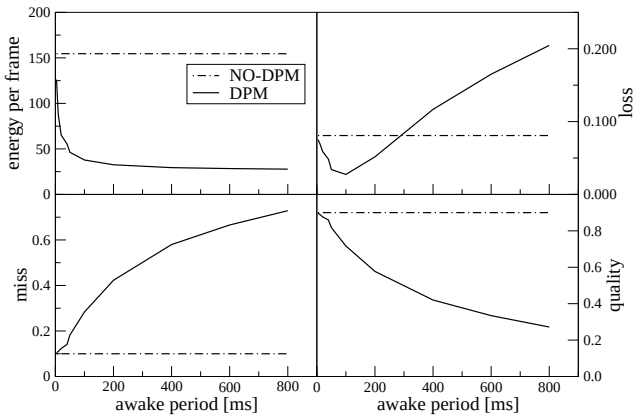

Figure 4. Performance comparison of the *streaming* benchmark with and without DPM for the Markovian model.

erage client processing time is 9.7 msec, the average client timeout is 2 msec, and the average DPM shutdown period varies between 0 and 25 msec.

Second, we have to define the performance measures of interest via reward structures. They are expressed in the companion language of Æmilia as follows:

```
MEASURE throughput IS
  ENABLED(C.process_result_packet)
    -> TRANS_REWARD(1);

MEASURE waiting_time IS
  ENABLED(C.monitor_waiting_client)
    -> STATE_REWARD(1);

MEASURE energy IS
  ENABLED(S.monitor_idle_server)
    -> STATE_REWARD(2)
  ENABLED(S.monitor_busy_server)
    -> STATE_REWARD(3)
  ENABLED(S.monitor_awaking_server)
    -> STATE_REWARD(2)
```

The results of the performance analysis conducted on the Markovian models of *rpc* are reported in the left-hand-side graphs of Fig. 3. Throughput, average waiting time, and energy per request are plotted as a function of the timeout used by the DPM to issue shutdown commands. The energy per request is obtained as the ratio of the energy to the throughput.

Dot-dashed lines refer to the system without DPM, while solid lines refers to the same system with DPM. As expected, the shorter the timeout, the larger the impact of DPM. The limiting situations are represented by a DPM that issues a shutdown command as soon as the server goes idle (timeout = 0) and by a DPM that never issues shutdown commands (timeout = ∞). In the first case the impact of DPM is maximum, while in the last case the DPM has no effect.

It is also worth noting that the DPM is never counterproductive in terms of energy, meaning that the additional energy required to wake the server up from the sleep state is compensated, on average, by the energy saved while sleeping. On the other hand, energy savings are always paid in terms of performance penalties (i.e., reduced throughput and increased waiting time), so that the DPM is not transparent to the client.

4.2. Application to *streaming*

As far as the value of the parameters for the Markovian models of *streaming* are concerned, we assume that the access point buffer size and the client buffer size is 10, the average server service time is 67 msec, the average packet propagation time is 4 msec, the packet loss probability is 0.02, the average NIC checking time is 5 msec, the average NIC awaking time is 15 msec, the average initial client delay time is 684 msec, the average client rendering time is 67 msec, the average DPM shutdown period is 5 msec, and the average DPM awake period varies between 0 and 800 msec.

For the *streaming* application we evaluate the impact of DPM based on four metrics expressed in the companion language of Æmilia: the average energy consumed by the NIC to receive each frame (energy per frame), the probability of losing a frame because of a buffer-full event (loss), the probability of violating a real-time constraint because of a buffer-empty event (miss), and the overall quality of service measured as the probability of delivering a video frame in time (quality).

The four measures are plotted in Fig. 4 as functions of the awake period of the PSP protocol. Solid and dot-dashed curves refer to the Markovian models with and without

DPM, respectively. As expected, the DPM has a larger impact for longer awake periods. In fact, the longer the awake period, the longer the sleep time of the NIC, regardless of the actual traffic. This has a beneficial impact on consumption and a negative effect on service quality. Interestingly, the loss rate is non monotonic. This can be explained by observing that the loss rate is the sum of the contributions of both buffers. When the NIC is in low-power doze mode, no frames can be received, thus filling up the AP buffer, while emptying the client-side buffer. For short-enough awake periods, the lower pressure on the client-side buffer overcomes the effect of the increased pressure on the AP buffer.

It is also worth noting that for awake periods between 0 and 100 msec, the energy per frame decreases much faster than the overall quality, while for awake periods above 100 msec the marginal energy savings become negligible at the cost of sizeable quality degradations. In practice, an awake period of 50 msec provides energy saving around 70% at the cost of a negligible impact on service quality.

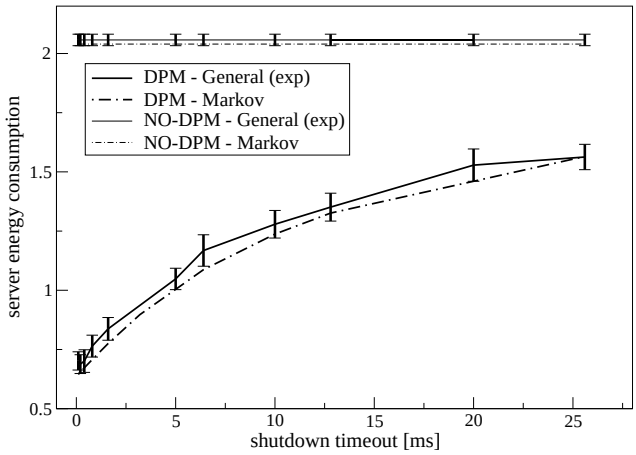

Figure 5. Validation of the general model of the *rpc* benchmark.

5. Comparing the General Models

The third model considered in the methodology of Fig. 1 is a general model obtained from the Markovian one by replacing exponentially distributed delays with generally distributed delays wherever necessary, thus resulting in a more realistic description of the system with and without DPM.

Replacing exponential distributions with general distributions may not be a smooth process. This is the case e.g. with Æmilia, as it does not directly support general distributions. The above replacement in Æmilia requires to switch from a continuous-time description to a discrete-time one regulated by a clock, in which the event occurrences are scheduled by sampling the corresponding action durations from general probability distributions. As a consequence, in general we need to guarantee some form of performance consistency between the general model and the Markovian model. This is accomplished by verifying that both models result in comparable values for the considered average performance measures when substituting exponential distributions for general distributions in the general model.

Besides such a performance-related validation, we observe that the replacement of exponential distributions with general distributions no longer having infinite support may alter the state space of the general model with respect to the state space of the Markovian model, hence the functional properties of the two models. As a consequence, in such a case the noninterference analysis must be repeated on the general model.

The general models with and without DPM can be simulated through standard techniques in order to estimate at a certain confidence level the same average performance measures of interest. In the specific case of Æmilia, the simulation parameters – run number and run length – are provided in an auxiliary specification, together with the reward-like description of the performance measures to be estimated.

Due to lack of space, in the following we leave out the Æmilia specifications of the general models of the two case studies, as well as the simulation-related auxiliary specifications. Such specifications can be found at `www.sti.uniurb.it/bernardo/twotowers/`

5.1. Validation

Since the general model for both case studies is obtained from the Markovian one via discretization, a consistency check is needed. Using TwoTowers we have carried out a parametric cross-validation by assigning to the general model exponential distributions consistent with the rates of the corresponding Markovian model. The general model has then been simulated and the results compared with those obtained by analyzing the Markovian model.

Due to lack of space, we show in Fig. 5 only the results for the *rpc* benchmark. This figure plots the estimates provided by the general and Markovian models for different values of the shutdown period. Simulation results have been obtained by averaging over 30 runs. Error bars are used in the plot to represent the 90% confidence intervals.

The good agreement between the values of the considered performance measure for the two models is apparent. On the other hand, the values are not identical mainly because of the discretization applied when introducing the general distributions.

5.2. Application to *rpc*

The values of the parameters of the general models of the *rpc* benchmark are similar to those of the corresponding Markovian models, with the difference that the server service time, the server awaking time, the client processing time, the client timeout, and the DPM shutdown period are now deterministic, while the the packet propagation time is normally distributed (with standard deviation 0.0345 msec).

Comparative simulation results with and without DPM are plotted in the right-hand-side graphs of Fig. 3 for different (deterministic) shutdown periods. First of all, we observe that there is a sizeable difference from the results obtained for the Markovian model (reported on the left-hand-side of the same figure). This difference motivates the application of a general specification to model systems characterized by non-exponential distributions.

The three parameters of interest have a bi-modal dependence on the shutdown timeout: for timeouts shorter than the average idle period (11.3 msec), the energy grows linearly with the timeout, while the waiting time and the throughput are constant; for timeouts larger than the average idle period, the DPM has no effect on the measured parameters. In a deterministic system, the transition between the two modes would be instantaneous. The smooth transition observed in Fig. 3 for timeout values around the average idle period is due to the normal probability distribution associated with the radio channels.

From the graphs we observe that: (i) the DPM is counterproductive if the timeout value is close to the actual idle period (in this case, in fact, the server needs to wake up right after a shutdown), (ii) energy savings provided for short timeouts are paid both in terms of waiting time and in terms of throughput, (iii) the DPM is transparent to the user in terms of performance only when it does not provide any energy saving.

5.3. Application to *streaming*

The parameters of the general models of the *streaming* application have been characterized based on the results of real-world measurements performed on a HP's IPAQ 3600 handheld PC with a CISCO AIRONET 350 [6] wireless NIC, accessing a remote server connected to a CISCO 350 Series access point [5], while the model for the radio channel is the same Gaussian model used for the *rpc* benchmark.

Simulation results are reported in Fig. 6. The dependency of the energy per frame from the awake period is quite similar to that obtained for the Markovian model (see Fig. 4 for comparison). There are, however, significant differences in the behavior of the performance metrics that make simulation results much more informative than Markovian analysis.

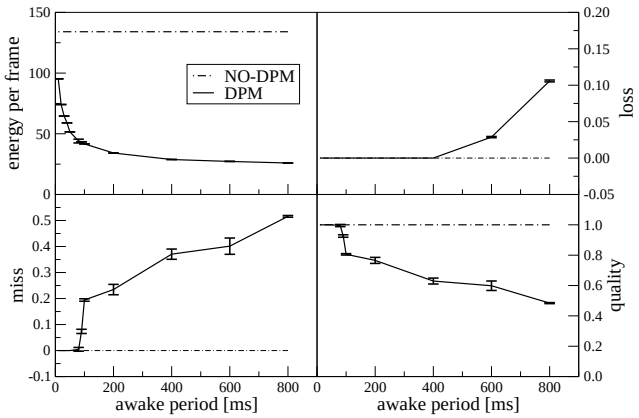

Figure 6. Performance comparison of the *streaming* benchmark with and without DPM for the general model.

In particular, there is no packet loss for awake periods up to 400 msec, and no packet miss for awake periods up to 100 msec. As a consequence, the quality of service perceived by the user is not affected by the presence of DPM as long as the awake period is below 100 msec. On the other hand, an awake period of 100 msec is sufficient to save more than 70% of the energy spent by the NIC.

Doubling the awake period from 100 msec to 200 msec reduces the quality of service below 0.8, with a negligible marginal improvement of the energy saving. Interestingly, the CISCO AIRONET 350 provides a choice between awake periods of 100 msec and 200 msec, making the comparison between the energy-quality tradeoff they provide of practical interest.

From our analysis, we conclude that a MAC-level DPM with 100 msec awake periods is transparent to the streaming video client.

6. Conclusion

In this paper we have proposed a formal method based incremental methodology for assessing the impact of DPM both on the functionality and the performance of the system in which the DPM is introduced. The methodology has then been applied to two realistic case studies.

From the functional viewpoint, it is worth remarking the suitability of the noninterference analysis for investigating the transparency of the DPM.

The results obtained from the performance comparison are summarized by the graphs of Fig. 7 and 8, showing the energy-quality tradeoff provided by the DPM for the two case studies when varying the DPM operation rates. Thin curves refer to Markovian models, while thick curves refer to general models. The difference between Markovian anal-

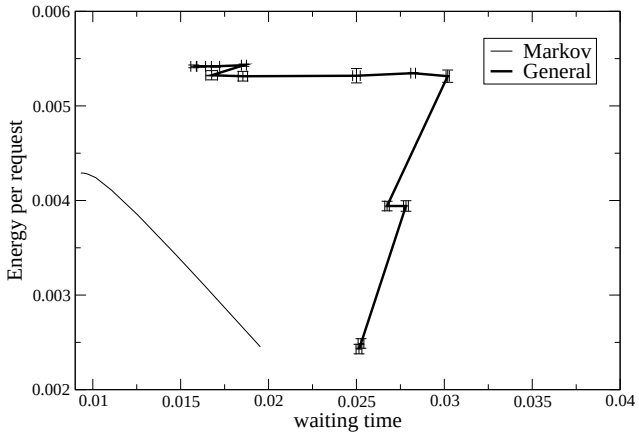

Figure 7. Tradeoff between energy consumption and waiting time for the Markovian and general *rpc* models.

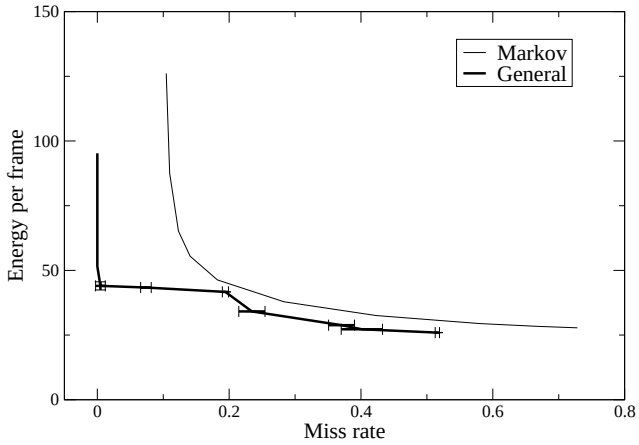

Figure 8. Tradeoff between energy consumption and miss rate for the Markovian and general *streaming* models.

ysis and general simulation is greater for the rpc server. It is also worth noting that many points of the tradeoff curve of the general *rpc* model are beyond the Pareto curve, since they are overcome by other points both in terms of energy efficiency and in terms of performance. The sub-optimal points correspond to the DPM timeout values close to the actual idle time, which make DPM counterproductive. For the tradeoff curve of the general model of the *streaming* application, we observe that sizeable energy savings can be obtained at no cost in terms of quality of service, making the DPM completely transparent to the user.

We finally remark that the tradeoff curves provided by Markovian and general models for the *streaming* application have the same qualitative behavior, even if the quantitative effect of the Markovian approximation is sizeable.

References

[1] L. Benini, A. Bogliolo, and G. De Micheli, *"A Survey of Design Techniques for System-Level Dynamic Power Management"*, in IEEE Trans. on VLSI Systems 8:299-316, 2000.

[2] M. Bernardo, *"TwoTowers 3.0: Enhancing Usability"*, in Proc. of the *11th IEEE/ACM Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2003)*, IEEE-CS Press, pp. 188-193, Orlando (FL), 2003.

[3] M. Bernardo and M. Bravetti, *"Performance Measure Sensitive Congruences for Markovian Process Algebras"*, in Theoretical Computer Science 290:117-160, 2003.

[4] M. Bernardo, L. Donatiello, and P. Ciancarini, *"Stochastic Process Algebra: From an Algebraic Formalism to an Architectural Description Language"*, in *Performance Evaluation of Complex Systems: Techniques and Tools*, LNCS 2459:236-260, 2002.

[5] Cisco System, *"Cisco Aironet 350 Series Access Points"*, http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/accsspts/index.htm, 2003.

[6] Cisco System, *"Cisco Aironet 350 Series Wireless LAN Adapters"*, http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/350cards/index.htm, 2003.

[7] R. Focardi and R. Gorrieri, *"A Classification of Security Properties"*, in Journal of Computer Security 3:5-33, 1995.

[8] J.A. Goguen and J. Meseguer, *"Security Policy and Security Models"*, in Proc. of the *Symp. on Security and Privacy (SSP 1982)*, IEEE-CS Press, pp. 11-20, 1982.

[9] R.K. Gupta, S. Irani, and S.K. Shukla, *"Formal Methods for Dynamic Power Management"*, tutorial at *IEEE/ACM Int. Conf. on Computer Aided Design (ICCAD 2003)*, ACM Press, 2003.

[10] LAN/MAN Standards Committee of the IEEE Computer Society, *"Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band"*, 1999.

[11] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989.

[12] G. Norman, D. Parker, M. Kwiatkowska, S.K. Shukla, and R.K. Gupta, *"Formal Analysis and Validation of Continuous-Time Markov Chain Based System Level Power Management Strategies"*, in Proc. of the *IEEE High-Level Design Validation and Test Workshop*, IEEE-CS Press, pp. 45-50, 2002.