Predicting the Impact of Dynamic Power Management *with Formal Methods*

Marco Bernardo

University of Urbino - Italy

Dynamic Power Management

- *Context:* designing battery-powered devices for mobile embedded systems.

- Major issue: reducing the power consumption of such devices (battery-powered).

- DPM: modify at run time the power consumption of the devices by changing their state or scaling their voltage/frequency.

- Several DPM schemes and policies proposed in the literature.

DPM Transparency

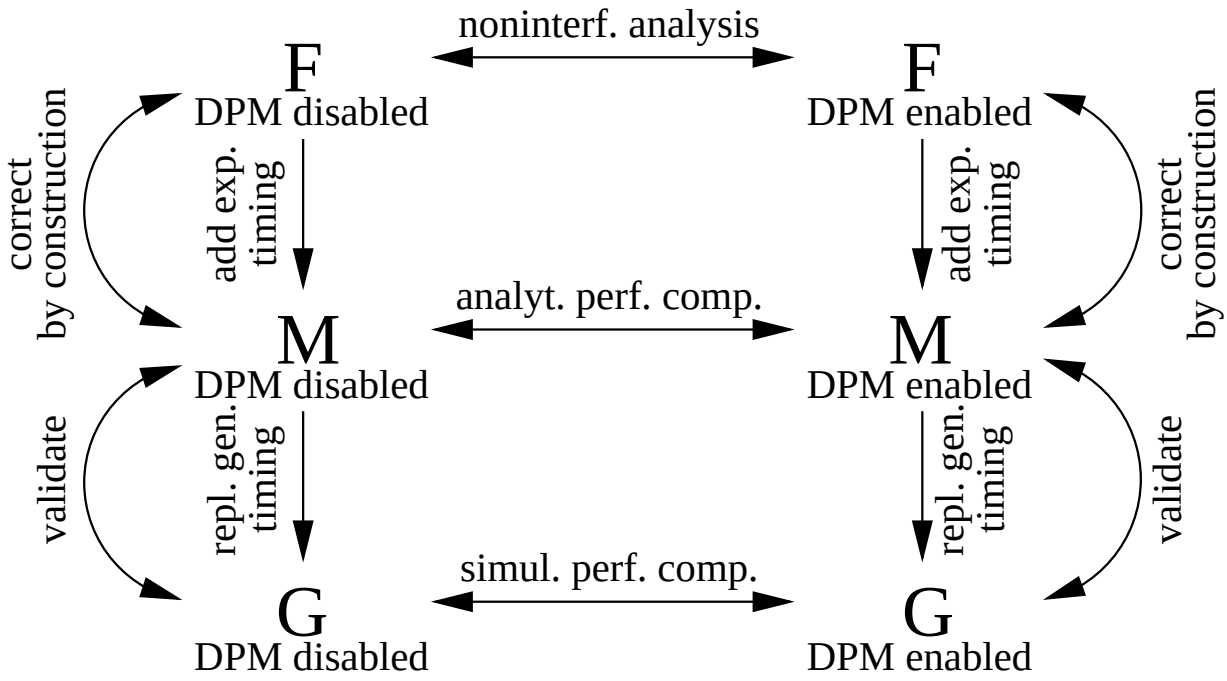
- *Problem:* if we introduce a DPM, what is the effect on the system functionality and performance?

- Make sure that the system behavior will not be significantly altered and that the quality of service will not go below an acceptable threshold.

- *Proposal:* incremental methodology based on formal methods to predict the DPM impact in the early design stages.

Predictive Methodology

- The DPM activities can be divided into two classes:
  - Activities that modify the state of the power-manageable device.
  - Activities that gather information about the state of the power-manageable device.
- We say that the DPM is enabled/disabled depending on whether it is capable of modifying the state of the power-manageable device or not.
- *Basic idea:* comparing the behavior and the efficiency for the system with the DPM disabled and the system with the DPM enabled.



- Three pairs of models of the system compared in three phases: functional, Markovian and general.

- Each model is incrementally obtained from the previous one by adding further details.

- Within each pair, one model refers to the system with the DPM disabled, while the other one refers to the system with the DPM enabled.

First Phase

- *Does the introduction of the DPM alter the system functionality?*

- Non-interference approach: the high enclave is not interfering with the low enclave if what the high one can do has no effect on what the low one can see.

- High: DPM state-modifying activities.

- Low: all activities carried out by the system users.

- Non-interference based on equivalence checking.

- Functional transparency: the functional model with the state-modifying activities of the DPM being made unobservable (DPM hidden) is equivalent to the functional model with the same activities being prevented from taking place (DPM disabled).

- Distinguishing modal logic formula in case of inequivalence.

- Diagnostic piece of information to guide the modification of the behavior of the DPM and/or the system.

Second Phase

- It is practically impossible that the introduction of the DPM does not alter the quality of the service delivered by the system.

- *Is it possible to achieve a trade-off between the energy consumption and the overall system efficiency?*

- Additional specification of the timing of each system activity (performance evaluation).

- Exponential delays turn the two functional models into two Markovian models.

- Functional transparency is automatically preserved (unless some DPM state-modifying activities are characterized through immediate transitions).

- Compare energy consumption, system throughput, radio channel utilization, and quality of service for the system with the DPM disabled and with the DPM enabled when varying the DPM operation rates.

- Standard techniques for the solution of reward Markov chains.

Third Phase

- *Are the two Markovian models realistic?*

- Replacing exponential distributions with general ones wherever necessary turns the two Markovian models into two general models.

- Validate the general models against the Markovian ones in case of major modifications (e.g. because the formalism does not support general distributions).

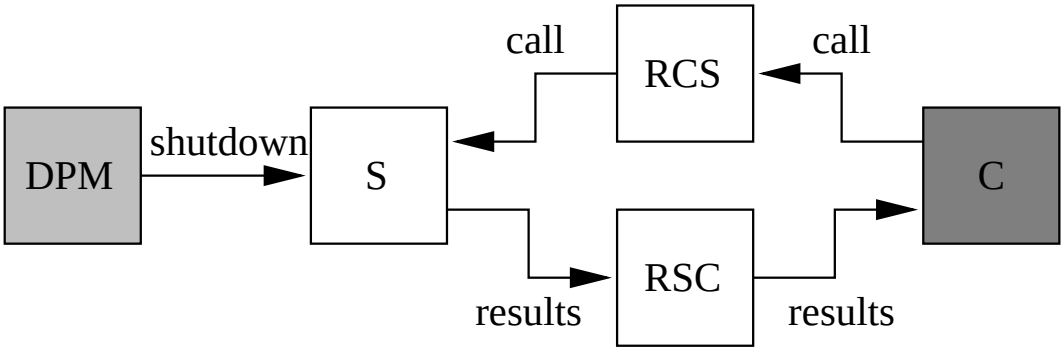
- Functional transparency is automatically preserved (unless the duration of some DPM state-modifying activities are characterized through general distributions with finite support).

- Standard simulation techniques to compare the same performance measures as the second phase.

- Useful to guide the decision about whether it is worth introducing the DPM in certain realistic scenarios and, if so, to tune the DPM operation rates without compromising the quality of service.

- Do not skip the second phase (smaller modeling gap, validation of the simulation results in the early design stages).

RPC Case Study



- Two perfect radio channels RCS and RSC (no packet loss).
- Naive blocking client C with no timeout mechanism.
- The server S has four power states:
  - Idle (waits for a call to arrive).
  - Busy (delivers the requested service).
  - Sleeping (after DPM shutdown command).
  - Awaking (after call arrival while sleeping).
- Trivial DPM sending shutdown commands independently of the current state of the server (idle/busy).
- Server always sensitive to DPM shutdown commands (call processing interruption).

RPC: Phase 1

- Æmilia functional model of the client:

```
ELEM_TYPE Client_Type(void)

 BEHAVIOR

  Client(void; void) =
   <send_rpc_packet, _> . <receive_result_packet, _> .
    <process_result_packet, _> . Client()

 INPUT_INTERACTIONS
  UNI receive_result_packet

 OUTPUT_INTERACTIONS
  UNI send_rpc_packet
```

- Æmilia functional model of the radio channels:

```
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
```

- Æmilia functional model of the server:

```
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
```

- Æmilia functional model of the DPM:

```
ELEM_TYPE DPM_Type(void)

 BEHAVIOR

  DPM_Beh(void; void) =
   <send_shutdown, _> . DPM_Beh()

 INPUT_INTERACTIONS
  void

 OUTPUT_INTERACTIONS
  UNI send_shutdown
```

- Æmilia description of the system topology:

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

ARCHI_INTERACTIONS
 void

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
```

- The non-interference analyzer of TwoTowers has been applied to the Æmilia description of the functional model.

- TwoTowers automatically produces the two different views of the system – DPM disabled and DPM enabled – when providing the auxiliary specification:

```
HIGH DPM.send_shutdown

LOW  C.send_rpc_packet;
     C.receive_result_packet;
     C.process_result_packet
```

- The two views are not equivalent.

- Modal logic formula explaining why the DPM interferes with the system functional behavior as perceived by the client:

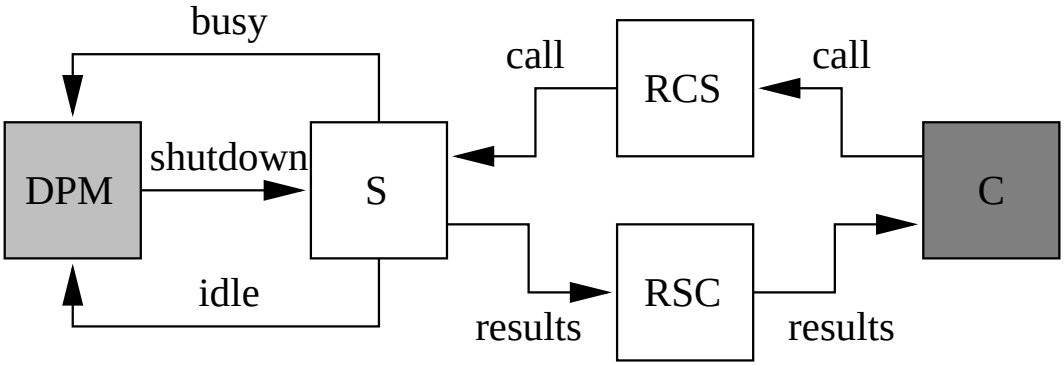
```
EXISTS_WEAK_TRANS(
  LABEL(C.send_rpc_packet#
        RCS.get_packet);
  REACHED_STATE_SAT(
    NOT(EXISTS_WEAK_TRANS(
          LABEL(RSC.deliver_packet#
                C.receive_result_packet);
          REACHED_STATE_SAT(TRUE)
        )
    )
  )
)
```

- If the DPM is enabled a deadlock can occur.

- After receiving a call, the server is shut down by the DPM.

- The server cannot wake up until it receives another call, but the client cannot send another call because it is blocking and does not use any timeout mechanism.

RPC Case Study Revisited



- Blocking client implementing a timeout mechanism.
- Realistic radio channels (may lose packets).
- Server informing the DPM about its idle/busy state.
- DPM prevented from shutting down the server if busy.

RPC Revisited: Phase 1

- Non-interference achieved: DPM functionally transparent from the client viewpoint.

RPC Revisited: Phases 2 and 3

- Values of the performance parameters:
  - Average server service time: 0.2 msec.
  - Average server awaking time: 3 msec.
  - Average packet propagation time: 0.8 msec.
  - Packet loss probability: 0.02.
  - Average client processing time: 9.7 msec.
  - Average client timeout: 2 msec.
  - Average DPM shutdown period: 0 to 25 msec.

- Reward-based performance measure specification:

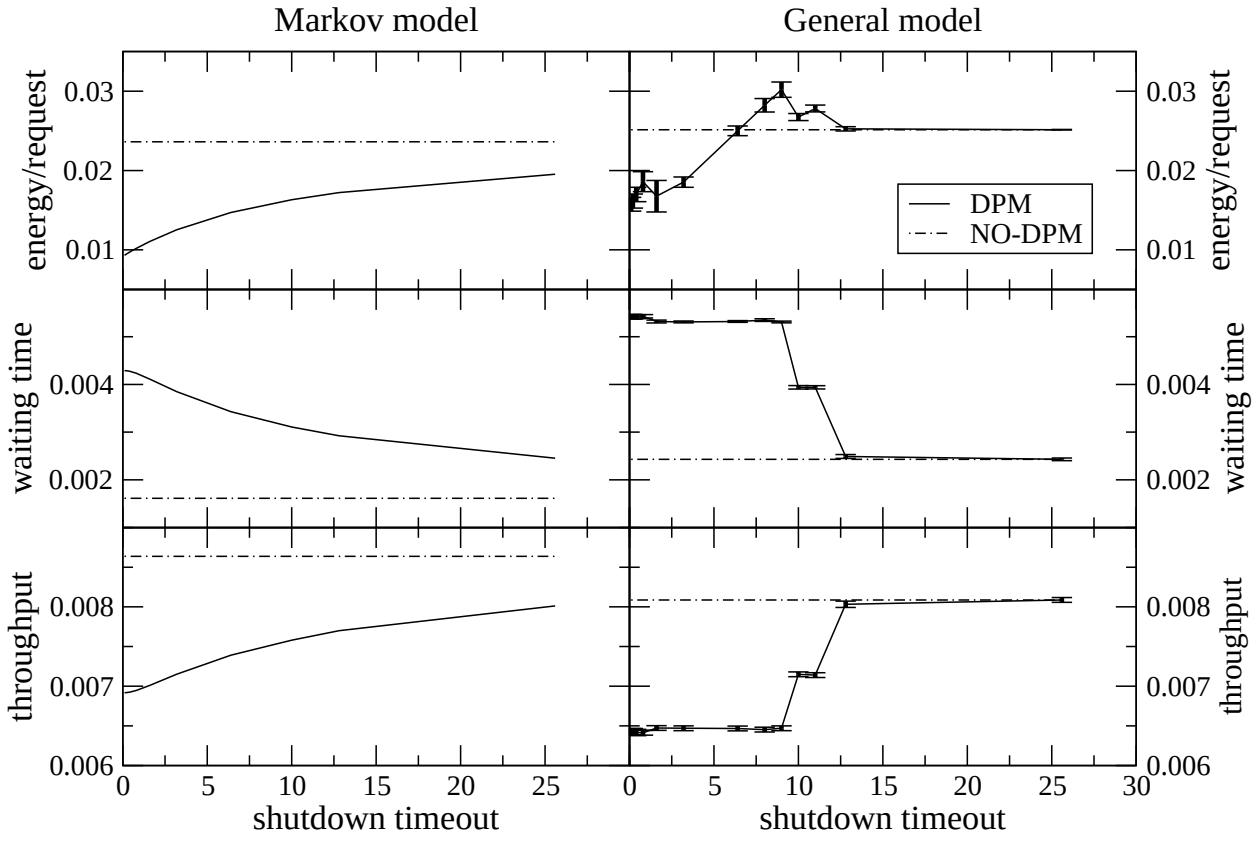
```
MEASURE throughput IS
 ENABLED(C.process_result_packet) -> TRANS_REWARD(1);

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)
```

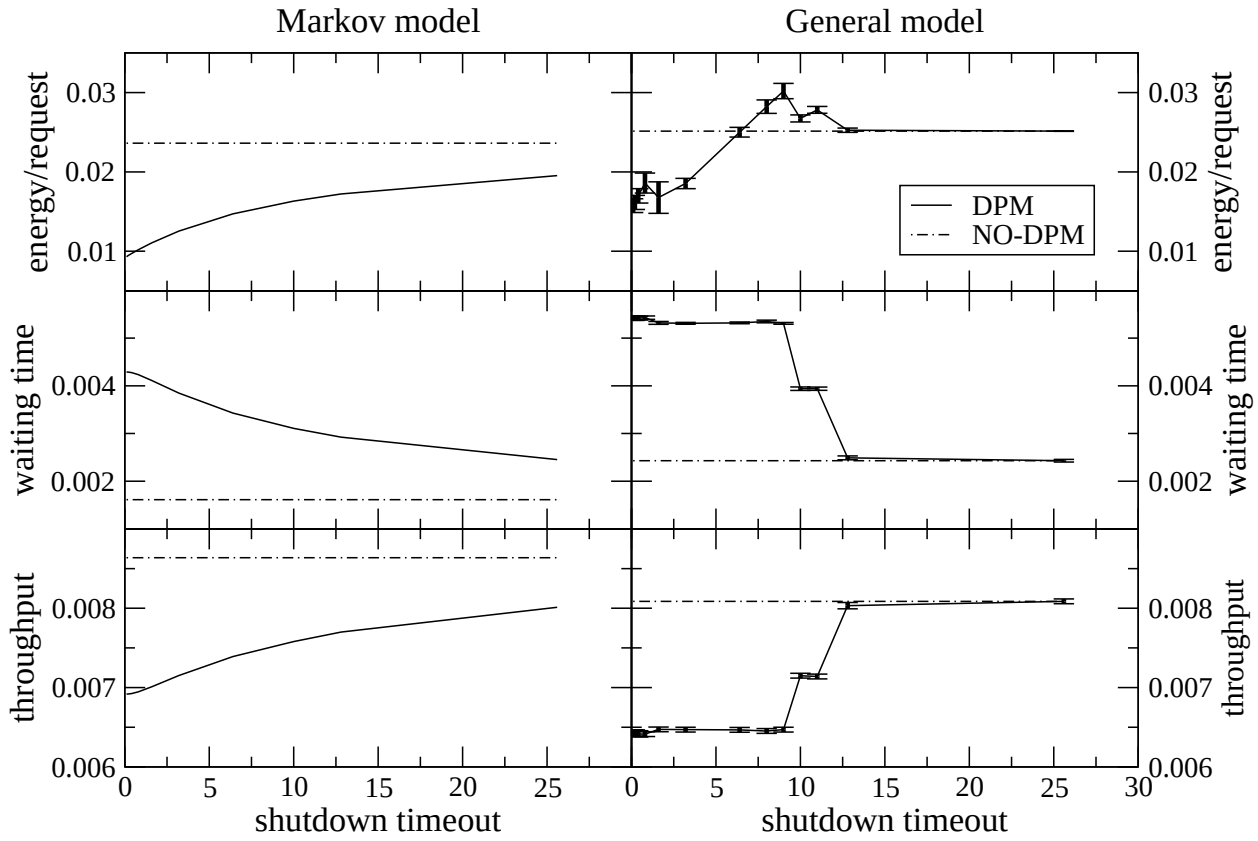
- For the third phase, additional specification of the simulation experiment details.

- Markovian comparison with TwoTowers:

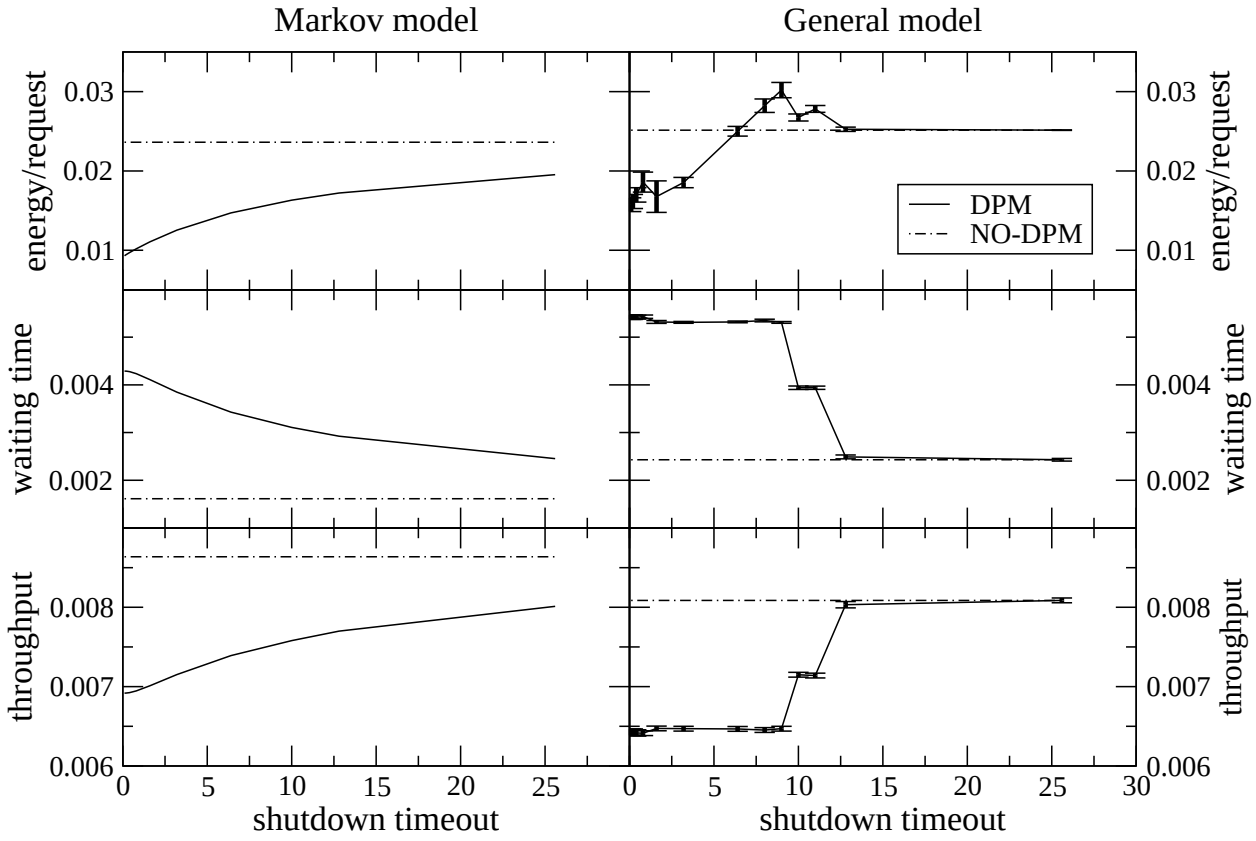


  - The shorter the period, the larger the impact.
  - DPM never counterproductive in terms of energy: the additional energy required to wake up the server from the sleep state is compensated on average by the energy saved while sleeping.
  - DPM not transparent to the client from the performance viewpoint: energy savings are always paid in terms of performance penalties (reduced throughput and increased waiting time).

- Simulative comparison with TwoTowers:



  - Packet propagation time normally distributed, with all the other time-related parameters being deterministic (no more exponentially distributed).
  - Sizeable difference with respect to the figures obtained from the Markovian models.
  - Good agreement when replacing general distributions with exponential ones (discretization).



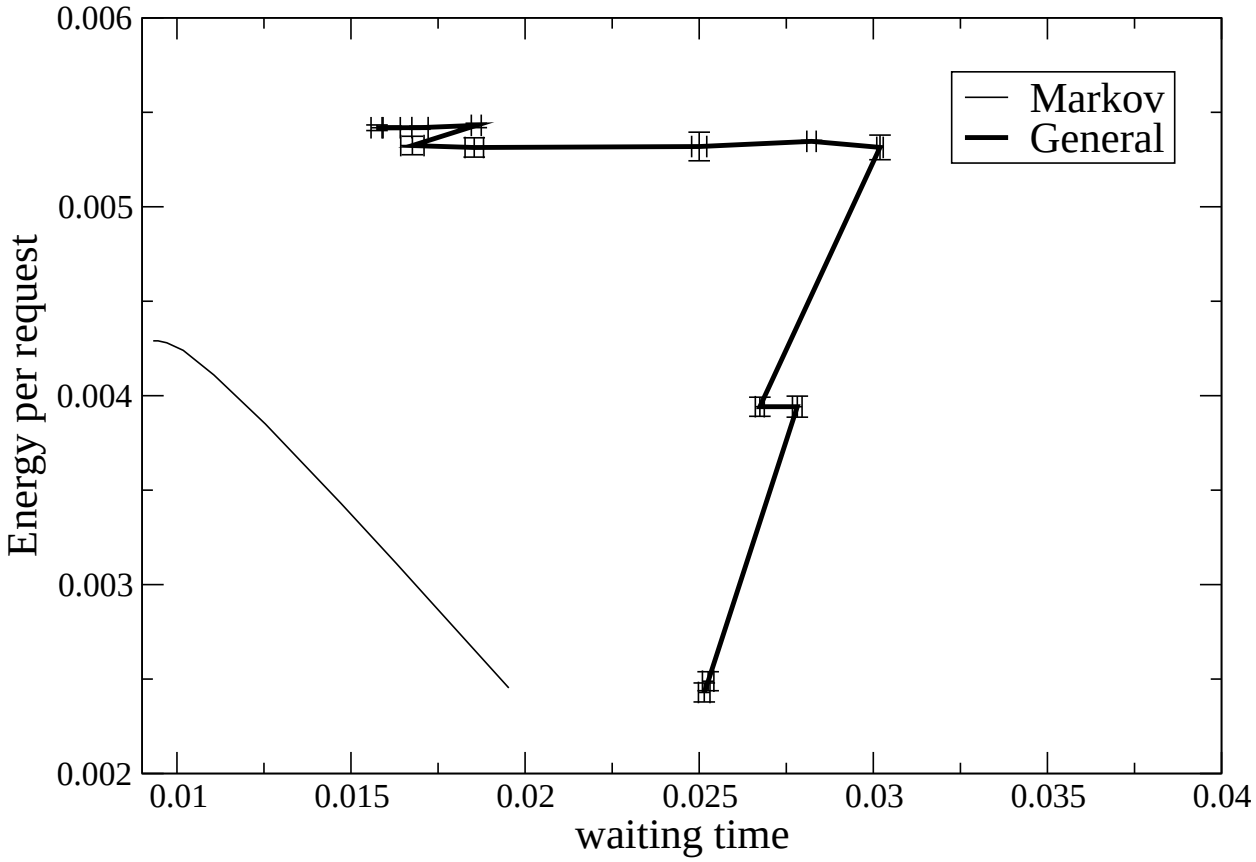
– Bi-modal dependence: if the shutdown period is shorter than the average idle time (11.3 msec), the energy grows linearly, while the waiting time and the throughput are constant; for larger periods, the DPM has no effect.

– DPM counterproductive if the shutdown period is close to the average idle time (the server needs to wake up right after a shutdown).

– Energy savings for short periods paid in terms of increased waiting time and reduced throughput.

– DPM performance transparent to the client only when providing no energy saving.

- Tradeoff curves:



  - Energy-quality tradeoff when varying the DPM shutdown period.
  - Optimal tradeoff in the Markovian case.
  - In the general case many points of the tradeoff curve are beyond the Pareto curve (values of the DPM period close to the average idle time).

Conclusion

- Incremental methodology to assist the design of mobile computing devices by predicting the DPM impact on the functionality and the performance.

- It can also be exploited for optimization purposes, by tuning the DPM operation parameters in order to achieve a satisfactory energy-quality tradeoff (if any).

- Based on formal methods, but independent of specific formal description techniques (as long as all the necessary ingredients are provided).

- Suitability of the non-interference analysis (detection of illegal information flows) for investigating the functional transparency of the DPM.