Journal of Low Power Electronics, Vol. 6, N° 2, August 2010

A Markovian Decision-based Approach for Extending the Lifetime of a Network of Battery-Powered Mobile Devices by Remote Processing

Peng Rong[1] and Massoud Pedram[2]

[1] Brocade Communications Systems, San Jose, CA, *prong2006@gmail.com*
[2] University of Southern California, Los Angeles, CA, *pedram@usc.edu*

* Corresponding Author:
Massoud Pedram
Dept. of Electrical Engineering
University of Southern California
Los Angeles CA 90089
Ofc: (213) 740-4458
Fax: (213) 740-9803
Email: pedram@usc.edu

Date of Receiving:

Date of Acceptance

A Markovian Decision-based Approach for Extending the Lifetime of a Network of Battery-Powered Mobile Devices by Remote Processing

Peng Rong and Massoud Pedram

Abstract – This paper addresses the problem of extending the lifetime of a battery-powered mobile host in a client-server wireless network by using task migration and remote processing. This problem is solved by first constructing a stochastic model of the client-server system based on the theory of continuous-time Markovian decision processes. Next the dynamic power management problem with task migration is formulated as a policy optimization problem and solved exactly by using a linear programming approach. Based on the off-line optimal policy derived in this way, an on-line adaptive policy is proposed, which dynamically monitors the channel conditions and the server behavior, takes into account real-time constraints, and adopts a client-side power management policy with task migration that results in optimum energy consumption in the client. Experimental results demonstrate that the proposed method outperforms existing heuristic methods by as much as 35% in terms of the overall energy savings.

Key Words – Client-server system; remote processing; network lifetime; Markovian decision processes.

1 INTRODUCTION

Extending the battery lifetime is one of the most critical and challenging problems in mobile battery-powered systems. Dynamic power management (DPM), which refers to a selective shut-off or slow-down of the idle or underutilized components, has proven to be a very effective technique in reducing power consumption of such systems. However, an implicit assumption in all of the previous DPM works (Benini et al. [1999] and Qiu et al. [2001]) is that local tasks of a mobile device are executed on the device itself. This is true if the mobile device has no communication capabilities with other mobile devices. However, when we consider a mobile host within a mobile network, which carries a wireless LAN card and can interchange data with other mobile hosts or fixed base stations over a wireless channel, the situation becomes quite different. A host with heavy workload may ask other hosts or the base station to help it reduce its workload by dispatching local tasks to these remote sites for processing. In this way, the mobile host may save power and extend its service lifetime.

Many key applications running on mobile platforms can benefit from task migration and remote processing. These applications include image processing, e.g., target detection and recognition used in robot control (Kremer et al. [2001]), voice recognition (Smailagic et al. [2002]), and large-scale numerical computations (Rudenko et al. [1998]).

The effectiveness of the remote processing technique is limited by the fact that data transmission over wireless channel results in additional power consumption. Energy savings on the local host is achieved only if the total energy for transmitting a task and receiving the result is less than the energy consumed for local execution of that same task. The rather large energy dissipation cost of wireless communication in mobile network of battery-powered devices makes the problem of deciding whether to execute a local task on the local host or to dispatch it to another mobile host for remote processing a very important one. In effect, energy-conscious policies must carefully consider the energy tradeoff between communication and computation and the task execution time from the viewpoint of the local host as well as the total energy dissipation for executing a task from the viewpoint of the network of mobile hosts.

A number of research results related to this problem have been reported in the literature. Experiments performed in Smailagic et al. [2002] and Rudenko et al. [1998] demonstrated the potential of remote processing for significant power savings in a number of real time tasks. The results of one experiment reported in reference Rudenko et al. [1998] are depicted in Figure 1. In this experiment, the authors compared the mobile computer's energy consumptions for local and remote execution of the Gaussian solution

of a system of linear equations. For remote execution, the entire coefficient matrix is shipped and the solution vector returned. It was observed that when the (linear equation) system size is around 500×500, the cost for moving the computation is close to the cost of local execution, but as the system size increases, the energy saving induced by remote execution becomes larger, e.g., energy savings is as large as 45% with a system size of 1000×1000. Based on CPU measurements, Othman et al. [1998] proposed an adaptive decision-making policy for a repetitive task. A remote processing framework was proposed in Rudenko et al. [1999], which supports process migration at the operating system level. This adaptive policy differs from that proposed in Othman et al. [1998] in that it can filter out the transient noise. Reference Kremer et al. [2001] proposed a compilation framework for remote processing, which can identify candidate remote computations within a single program. Unfortunately, these works do not consider any timing constraints on the tasks and assume that the user must be able to cope with any level of additional delay that may be introduced by remote processing. This limitation makes these techniques unsuitable for real-time applications, where violation of timing constraints may cause unacceptable loss in quality of service.

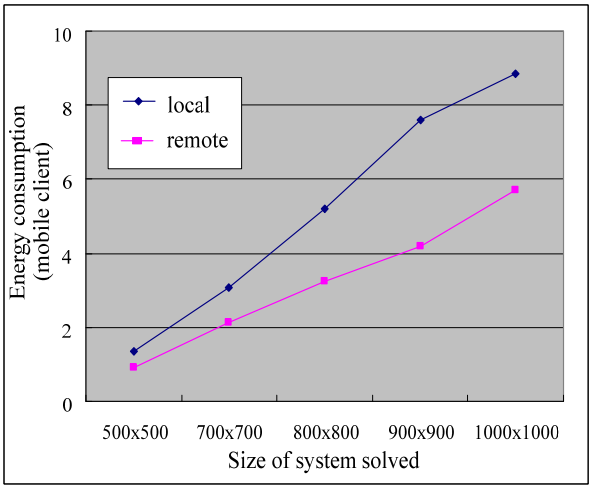

Figure 1. Power savings for remote execution of Gaussian solution of a system of linear algebraic equations.

IEEE 802.11 protocol supports two types of mobile networks: peer-to-peer architecture (ad-hoc mode) and client-server architecture (infra-structure mode). In ad-hoc network, there is no base station and communication among mobile hosts takes place without the need for a base station. In this case, the major issue is to balance the remaining energy resources of all mobile hosts so as to maximize the ad-hoc network

lifetime. This problem – although interesting - is different from the problem that we are addressing here for the infra-structure mode and is beyond the scope of the present paper. This paper targets a mobile device providing real time services in a client-server wireless network. The mobile battery-powered device (client) can communicate with and possibly migrate tasks to the "wall-powered" base station (server). Note that a client's tasks may in general be heterogeneous. Indeed, in the proposed framework, tasks are statistically modeled by parameters describing their expected arrival rate and service time on the client, and expected migration time to the server.

The article first presents a new Markovian Decision Process-based DPM framework for such a network. The proposed stochastic model is used for minimizing the power consumption of the mobile host by using remote processing while meeting real-time constraints. A preliminary version of this paper was published in the Proceedings of the 40th *Design Automation Conferenc*e. Compared to the conference version, this article has the following additional content:

1. Detailed derivations and discussion of key results
2. Improvements to the Server Model and a flow diagram of the RPR acceptance and execution on the Server
3. Changes to the problem formulation for the offline policy including significant revision of the formulation of the Delay Constraint, which enhances the optimality of the result by eliminating the empirically selected parameter D_c
4. Extension of the online policy to the G/G/1 server model
5. Discussion of why and where to add a security-enhancement state to the Client Model in order to protect migrated tasks from being eavesdropped or altered.

The remainder of this article is organized as follows. Related work and background are provided in Section 2. In Section 3, details of the proposed DPM framework are described. In Section 4, stochastic models of the client, the wireless channel, and the server are provided. In Section 5, the energy optimization problem is formulated as a mathematical program and two DPM policies are presented. Experimental results and conclusions are given in Sections 6 and 7, respectively.

2. BACKGROUND

Research on wireless communication has demonstrated that the multi-path fading and shadowing (slow fading) effects in a wireless channel may significantly degrade the signal-to-noise ratio, increase the error rate, and thus cause a large amount of delay and energy consumption for re-transmitting the corrupted packets. So when determining an

optimal policy for the client, a detailed and accurate model of a wireless channel should be constructed and used.

A (controllable) continuous-time Markov decision process, c.f. Feinberg et al. [2002], (CTMDP) is defined with a discrete state space; a generator matrix, where an entry represents the transition rate from one state to another; an action set; and a reward function. In CTMDP, the generator matrix is a parameterized matrix that depends on the selected action. An irreducible CTMDP has a unique limiting distribution that is independent of the initial conditions.

A complete power-managed system may comprise of different components, each with its own functionality and purpose. A simple example of one such system is depicted in Figure 2. Here, the system consists of three components: service requestor, service queue and service provider. The power manager gathers state information from the service requestor and service queue, and also controls the behavior of the service provider based on the utilized policy in order to reduce the overall energy consumption. To model this power managed system, each component is first modeled by a CTMDP. Next the state set of the complete system is obtained as the Cartesian product of the state set of each component minus the set of invalid states. By using the method of Qiu et al. [2001], the generator matrix of the whole system can be generated from the generator matrices of its components by using the tensor sum and/or product operations.

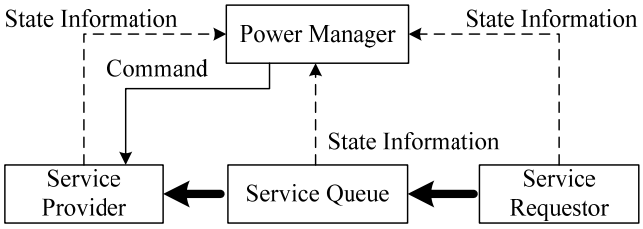

Figure 2. Simplified model of a power-managed system.

To increase the readability of the paper, we provide a summary of all key notation and definitions next.

Notation	Definition
D	A user-specified upper bound on the average task delay
T_h	A user-specified upper bound on the percentage of task requests which are denied by the service provider. This is also equal to the *task loss rate*.
λ_c	Average rate of incoming tasks to the mobile client
$1/\mu_c$	Average service time of a task on the mobile client

$1/\mu_m$	Task migration time for remotely executed tasks
w_i	State *i* of the wireless channel
PER_i	Packet error rate in w_i
$v_{i,j}$	Transition rate from w_i to w_j
$1/\mu_s$	Average service time of a task on the server
$\lambda_{s,i}$	Task incoming rate to the server in server-task generation state *i*
$P_{reject,i}$	Rejection probability of the client's remote process request (RPR) by the Server in server-task generation state *i*
k	Ratio of the processing speed of the mobile client to the server
c	Slack factor i.e., a constant factor by which the service time bound that is assigned to an RPR is larger than its execution time on the server
x	State of the whole power-managed system
a_x	An action enabled in state *x*
$f_x^{a_x}$	Frequency that state *x* is entered in and action a_x is chosen
$\gamma_x^{a_x}$	Expected cost, i.e. energy consumed when the system is in state *x* and action a_x is chosen
$P_{x,x'}^{a_x}$	Probability that the next state of the power-managed system is *x'* when its present state is *x* and action a_x is chosen
$\sigma_{x,x'}^{a_x}$	Transition rate of the power-managed system from state *x* to state *x'* when a_x is chosen
$ene(x,x')$	Energy associated with a successful transition from state *x* to state *x'*
$\tau_x^{a_x}$	Expected length of time that the system will stay in state *x* when action a_x is chosen
$f_x^{CM\|CC}$	Frequency of a transition whereby the CP enters state CM from a system state x where the CP was in state CC
$f_x^{CI\|CC}$	Frequency of a transition whereby the CP enters state CI from a system state *x* where the CP was in state CC
h_x	Variable used in the second order cone programming formulation of the nonlinear optimization problem

3. DPM FRAMEWORK

The proposed DPM framework consists of three major components: the clients (mobile hosts), a server (base station) and a wireless channel which carries the

communication packets between the client and the server. It is assumed that the server is AC-powered and has a much larger computational capability than the client. We also assume that the client services only its own local tasks and receives no request for remote processing from the server. This is a reasonable assumption since the AC-powered high-performance server is much more powerful from a processing point of view and has no energy limit, and thus it will execute its own tasks (in addition, it will execute tasks sent to it by the mobile hosts.) This also means that the server has all the hardware and software resources required to execute the tasks that are sent to it by the remote clients. Furthermore, for the same reasons, the server does not turn down any request for remote processing.

When a client desires to execute a task on the server, it sends a remote process request (RPR) to the server with a required real time constraint. Because the server may be busy executing other tasks (some local, other remote tasks that have previously arrived), it may reject the RPR from the mobile host because it may have determined that it cannot meet the required time constraint for the remote task. This is the only case in which the server rejects an RPR, that is, the server never turns down an RPR for reasons of server-side energy saving. When the RPR is rejected by the server, the client will have to perform the task locally. However, at that point, the client has wasted some valuable resources (energy and time) trying to migrate a task to the server and because it has failed in doing so, it still has to perform the requested task locally. It is therefore essential for the energy efficiency of the client and for its performance to minimize the probability for its RPRs to be rejected by the server.

The procedure/protocol for remote processing is explained next.

1. Based on estimations of the cost of executing a task locally and the task rejection ratio by the server, the client decides to migrate the task to the server. This task is called a remote execution candidate (REC). The client calculates the timing constraint for the execution of the REC.
2. The client sends an RPR to the server containing workload and timing constraint information about the REC.
3. When the server receives the RPR, it checks the status of the tasks waiting on the server to see whether the timing constraint for the REC can be satisfied. If so, the server will accept the application, otherwise, it will reject the application. (The server-side decision algorithm is explained in details in Section 4.3.) Whether or not the application is accepted, the server will inform the client of its decision by sending an acknowledgment (ACK) back to the client. Included in the ACK response are the

decision to accept or reject the RPR, and current status information about the server, i.e., the average incoming request rate and the average execution time of the tasks on the server side.

4. If the client receives a positive (acceptance) response from the server, then it will start to migrate the REC to the server. Otherwise, the client will proceed to execute the task locally.
5. When the client finishes the task migration step, it can immediately start processing a new task if one has arrived.
6. When the server completes the task, it will store the result in its own memory and immediately inform the client that the computation result is ready by sending a task done (DONE) message to the client.
7. If the client receives the DONE message from the server, then it will immediately contact the server and collect the computation result (RES). If the server does not see any activity from the client, then it will resend the DONE message at the next conference time. At that time, the client is guaranteed to be awake and therefore will receive the DONE message and will pick up the RES from the server. At the same time, if the client does not receive any message from the server and has not had a conference with the server since the last REC was sent off, then before the deadline for REC is expired, it will contact the server to pick up the RES. [1]
8. During the process of RPR negotiation (steps 1-3), REC handoff (step 4,5), and RES computation (step 6), and RES delivery (step 7), the client counts the number of packets that had to be re-transmitted due to unrecoverable errors in the received packets. This information will enable client to determine the wireless channel condition in real time.

4. MODELING

Because this paper focuses on the client-server architecture (i.e., the infra-structure mode of the IEEE 802.11b), we can assume that the mobile hosts (clients) in the network are independent of each other[2] and therefore when a client learns about the status of the

[1] This case actually means that the server finished the RES computation, send a DONE message to the client who was asleep and thus missed the server ACK.

[2] Other mobile hosts affect the target mobile host only because of packet collision in the wireless channel. In this paper, we treat this collision effect (which is transparently handled and minimized by MAC layer) as noise in the wireless channel.

server, it has all that it needs to make local decisions as to how it can improve its energy efficiency and thereby extend its battery lifetime. The client-server system is thus modeled by a joint CTMDP model, which is composed of CTMDP models of only three components: a single client, a wireless channel, and the server.

4.1 Model of the Client

We consider a (mobile) client that is continuously executing some real-time service processes for each incoming task. The QoS requirements for the client service are: 1) the average task delay is less than a predetermined value D; and 2) the task loss rate is less than a threshold T_h. Different tasks differ in the task size which is exponentially distributed. It is assumed that the relationships between the task size and its execution time on the client and the migration time over an error-free wireless channel are known in advance (for example, through profiling).

The model of the client is illustrated in Figure 3. It has three processors: Service Provider (SP), Conference Processor (CP), and Issue Processor (IP).

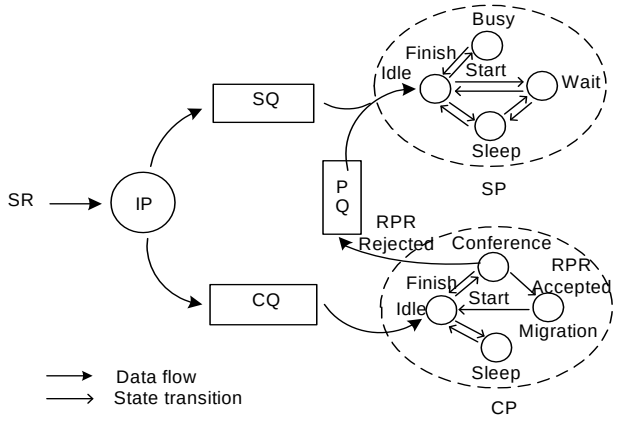

Figure 3. CTMDP model of a client.

The SP represents the component of a mobile device that can provide service for the service requests (SR) (e.g. the CPU). The CP is in charge of negotiation with the server for remote processing and task migration. When an REC is selected, the CP first sends a request for remote processing to the server, which includes the basic information about the REC, such as its expected computational workload and the relevant timing constraint. When receiving an RPR, the server checks its own resources and workload to see whether or not it can finish the task in the required time. If the timing constraint can be met, the request is accepted; otherwise, it is rejected. If the CP receives an "Accept" response from the server, it starts to send off (migrate) the task to the server. After

completing the task migration, the CP can immediately start a new negotiation with the server for the next REC. When the server finishes the required job, it stores the RES in its own memory and waits for the client to get them back. If the CP receives a "Reject" response, it moves the rejected REC out of the Conference Queue (CQ) and inserts it into the Priority Queue (PQ). The tasks in the PQ have a higher priority in receiving service from the SP compared to other tasks in the normal Service Queue (SQ). This makes sense because these tasks have already been held back because of the "failed" attempt to migrate them to the server. A typical CP is a WLAN Card with Direct Memory Access (DMA) capability. Since it can transmit and receive data with very little CPU intervention, we assume that the CP and the SP can work independently of one another. When a new task is generated, the IP decides whether to service it locally or make it a REC, and therefore insert the incoming task into the SQ or CQ, respectively. The IP is a low complexity and power-efficient processor (e.g., a PIC processor). We assume its power consumption can be neglected in comparison to the SP and the CP. The IP is always awake waiting for the arriving tasks and deciding whether to treat them as local or REC's. [3]

The definitions of the states of the SP are as follows.

Busy (SB): a working state, where the SP service the tasks waiting in the SQ or CQ.

Idle (SI): a full-power but non-functional state, during which the Power Manager (PM) may issue any of the following commands to the SP: Go-to-Busy, Go-to-Wait, Go-to-Sleep, Stay-in-Idle.

Wait (SW) and **Sleep (SS)**: low power states. The SP in the Wait state has a higher power consumption compared to the Sleep state, but its transition to Busy or Idle state requires more time and energy.

The detailed states of the CP are explained as follows:

Idle (CI): State reached when an RPR negotiation is concluded with a "Reject" response, or when the RPR is accepted by the server and the client has completed the task

[3] In this model, we assume all incoming tasks to the mobile client have the same priority. This is true when all tasks have similar real-time requirements. However, this model can be extended to handle multiple task priorities. In such a case, a set of SQ, CQ and PQ can be added in parallel for each priority class. Different timing constraints may be imposed to each queue set to reflect different real-time requirements. At the same time, precedence rules for task execution can be applied to these queues in order to ensure that tasks with higher priority are always executed first. A similar method has been adopted in reference Qiu et al. [2001].

migration step. It is also the state in which the CP receives commands from the PM to determine whether to start a new negotiation, go to sleep, or stay in idle.

Conference (CC): In this state, the client sends the RPRs to the server, waiting for a server response indicating acceptance or rejection of the current RPR. If the request is rejected, the CP goes to the Idle state and the REC is fetched out of the CQ and inserted into the PQ. If the REC is accepted, the CP goes to the Migration state.

Migration (CM): This state is reached after an RPR is accepted by the server. In this state, the client sends all the data necessary for performing the task to the server through the wireless channel. When the data-sending process is concluded, the CP goes back to the Idle state and at the same time the migrated task is removed from the CQ.

Sleep (CS): State reached when the PM decides to put the CP into the lowest power mode to save energy. In this state the front-end of the wireless LAN card is turned off, thus no communication from the server can be received.

All of the state transitions of the CP are assumed to be either exponentially distributed (e.g., the transition from the Migration state to the Idle state) or instantaneous (the only case is for the transition from the Idle state to the Conference state.)

It is worth noting that in the CP model, we do not explicitly create a state for receiving the data RES of an RPR that has been serviced by the server. The reason is twofold: i) The remote processing protocol/procedure described previously guarantees the transmission of computation RES from the server to the client; ii) It is more convenient from a modeling point of view to account for the time and power consumption overhead of receiving the data RES of an RPR in the Migration state.

Another point worth mentioning is that moving tasks from the mobile client to the remote server may incur security and trust issues. Since addition of security-related features to an RPR is only required after it is accepted by the server and immediately before the task migration is commenced, one can simply add another state to the CP model to account for this activity. Referring back to Figure 3, the "security enhancement" state (SE) should be put between the CC state and the CM state. The "RPR Accepted" edge will be directed from the CC state to the SE state, and there will be an unlabeled edge from the SE state to the CM state. Adding security features to the tasks causes additional energy and timing overheads. In this paper, for simplicity, we do not include the SE state in our model although its inclusion is straight-forward as described above.

4.2 Model of the Wireless Channel

The Markovian chain model has proven to be a very successful mathematical tool to describe a wireless channel. A lot of Markovian chain based models have been proposed, from two-state "Gilbert Elliot" model (Elliot [1963]) to hierarchical hidden Markov model (Yang et al. [2002]). Complex models work better in terms of capturing the higher order statistics of the wireless channel, but result in a nearly exponential increase in model complexity (Haggstrom [2002]). A real wireless channel is usually exposed to both fast and slow fading effects. The study in Zorzi et al. [1998] suggests that a two-state Markov chain model is quite accurate and remains insensitive to different coding/modulation schemes when the fading is slow, whereas independent, identically distributed (i.i.d.) processes are suitable for describing the fast fading effects. Based on this study and other similar published results, in this paper, we adopt a two-state continuous-time Markov process to model the slow fading effect. We assign a constant packet error rate *PER* to each state. These rates represent the expected packet error rate of the i.i.d. processes for describing the fast fading effect. The wireless channel model is shown in Figure 4, where $1/v_{1,2}$ and $1/v_{2,1}$ represent the expectation time that the wireless channel remains in state w_1 and w_2, respectively. Notice that it is straightforward to extend the two-state model to a higher-order model with more states to achieve higher accuracy, but a two-state wireless channel model is sufficient for our purpose.

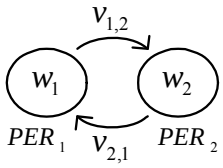

Figure 4. Two-state CTMDP model of wireless channel.

Let's define the average packet error rate (*PER, 0≤ PER≤1)* as the ratio of the number of un-recoverable packets, in spite of error-correction techniques such as CRC coding, to the total number of packets. We assume that any packet that is corrupted during transmission and for which error correction circuitry on the receiver side cannot fix the error must be re-transmitted. Let *t* denote the time required for transmitting an *n*-packet data over an error-free wireless channel. The expected time t_a for transmitting the same data over an error-prone wireless channel can be calculated as follows:

$$t_a = n \cdot \sum_{m=0}^{\infty} t_0 \cdot PER^m = \frac{nt_0}{1 - PER} = \frac{t}{1 - PER},$$

where t_0 denote the time for transmitting a single packet over an error-free wireless channel, and *m* is the number of re-transmissions.

4.3 Model of the Server

The server can be represented as an infinite M/M/1 queue (Dshalalow [1997]) with a multi-state task generator as shown in the Figure 5. Typically a server connects to a number of clients and has to perform a large amount of local computations. Therefore, we assume that under the stationary state condition: i) the rate of incoming tasks for the server is independent of any particular client; and ii) this rate changes slowly. From the client's viewpoint, what is important is the rejection probability of its RPRs. Thus we can reduce the order of the model as explained below.

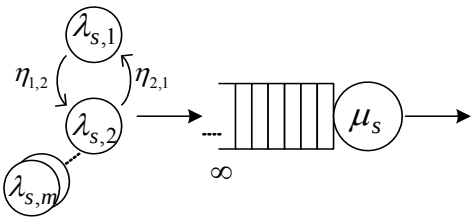

Figure 5. Queuing model of the server.

Let the aggregated incoming task rate of the server be λ_s and its average service time $1/\mu_s$. We enforce the condition: $\lambda_s<\mu_s$; otherwise, there will exist no limiting distribution for this queuing process. The limiting probability that the number of waiting tasks in the server queue equals n, is computed as:

$$p_n = \left(\frac{\lambda_s}{\mu_s}\right)^n \left(1 - \frac{\lambda_s}{\mu_s}\right).$$

First, we assume that the timing bound imposed on an RPR that is sent to the server for remote execution (i.e., the *timing bound*, RTB, for an RPR) is some *slack factor* $c \geq 1$ times larger than its actual execution time on the server. Next, we assume that the server uses a preemptive priority task queue with a *block list* to buffer the tasks waiting to be executed. When an RPR is received, the server compares its RTB and the total execution time of all tasks that it must perform, including those in the task queue and in the block list, to check whether this RTB can be satisfied if the RPR is executed on the server immediately after all currently scheduled tasks are completed. If this condition is met, then the RPR will be accepted; otherwise, it will be rejected. When an RPR is accepted by the server, it is inserted into the block list and assigned a priority number equal to its acceptance time (this algorithm is different from the earliest deadline first scheduling.) When the complete data (i.e., body) for this RPR is received by the server, it is moved into the task queue. In the task queue, the task having the least priority number will

preempt the currently executed task and will be serviced immediately. The flow diagram of RPR acceptance and execution on the server is shown in Figure 6. Based on this scheme, it is guaranteed that the time that an accepted RPR spends on the server will never exceed its assigned RTB. Using RTB with this scheme provides the mobile client an adjustable mechanism to bound the average time that its RPRs will be spending on the server.

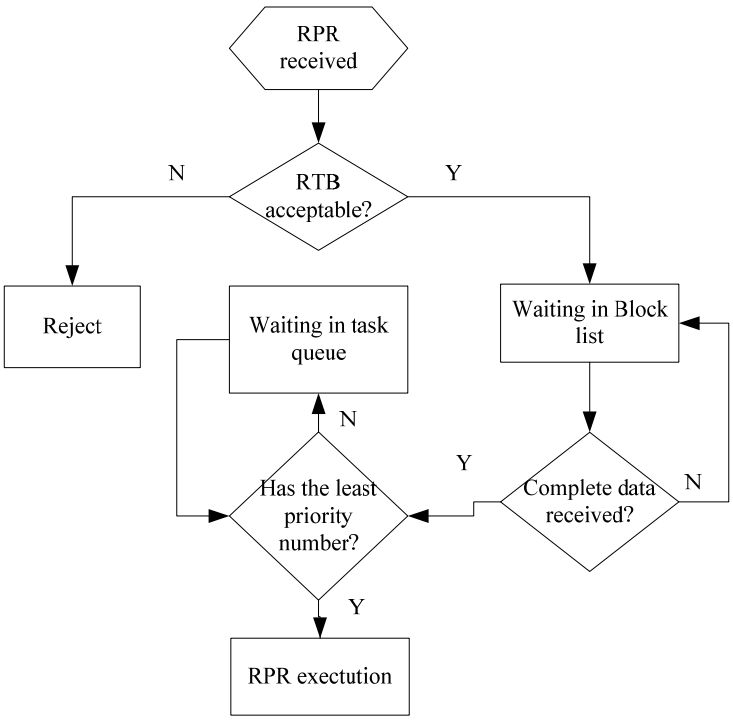

Figure 6. Flow diagram of RPR acceptance and execution on the server.

The execution time of an RPR on the server can be approximated by an exponential distribution with a mean value k/μ_c, where $1/\mu_c$ is average service time of the client, k is the ratio of processing speed of the client to the server, $k \leq 1$. So the RPR rejection probability is calculated as:

$$p_{reject} = f(\lambda_s, \mu_s) = 1 - \sum_{n=0}^{\infty} p_n \cdot \Pr\{(c-1)t_s \geq t_w \mid n\} = 1 - \sum_{n=0}^{\infty} p_n \cdot \int_0^{\infty} g_{w,n}(t_w)\, dt_w \cdot \int_{\frac{t_w}{c-1}}^{\infty} g_s(t_s)\, dt_s$$

$$= 1 - \sum_{n=0}^{\infty} p_n \cdot \int_0^{\infty} LT^{-1}\left[\left(\frac{\mu_s}{\mu_s + s}\right)^n\right] \exp\left(-\frac{\mu_c t_w}{k(c-1)}\right) dt_w = 1 - \sum_{n=0}^{\infty} p_n \cdot \mu_s^{\,n} \Big/ \left(\mu_s + \frac{\mu_c}{k(c-1)}\right)^n$$

$$= \frac{\lambda_s}{\mu_s} \cdot \frac{\mu_c}{(\mu_s - \lambda_s)k(c-1) + \mu_c} \qquad c \geq 1 \qquad (4\text{-}1)$$

where t_s is the execution time of the RPR on the server and t_w denotes the waiting time of the RPR on the server. $g_s(\cdot)$ and $g_w(\cdot)$ are the probability density functions of random variables t_s and t_w, respectively. $LT^{-1}[\cdot]$ represents the reverse Laplace Transform. By using a similar approach, for a given c, we can derive the average waiting time of an RPR on the server T_w, which is calculated as:

$$T_w = \sum_{n=0}^{\infty} p_n \cdot E\{t_w \mid t_w \le (c-1)t_s, n\} = \sum_{n=0}^{\infty} p_n \cdot \frac{\int_0^{\infty} t_w g_{w,n}(t_w)\,dt_w \cdot \int_{\frac{t_w}{c-1}}^{\infty} g_s(t_s)\,dt_s}{\Pr\{t_w \le (c-1)t_s \mid n\}}$$
$$= \frac{\lambda_s}{\mu_s - \lambda_s} \frac{k(c-1)}{\mu_s k(c-1) + \mu_c} \qquad c \ge 1 \tag{4-2}$$

Assume the average service time $1/\mu_s$ is constant. Thus, among the server parameters, only the incoming task rate λ_s is related to the rejection probability. Consequently, the server model may be reduced to a multi-state Markovian process as illustrated in Figure 7. In this model, it is assumed that the values of μ_c and k are constant.

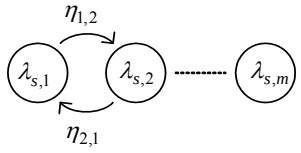

Figure 7. CTMDP model of the server.

5. POLICY OPTIMIZATION

We describe two policies: an *off-line optimal policy* and an *on-line adaptive policy*. The off-line optimal policy is computed based on the joint stochastic model of the client, the wireless channel and the server by using a linear programming approach. If the key characteristics of the wireless channel and the server are stable, then using the offline policy will result in the optimum energy saving solution. In practice, however, the channel conditions and the server workloads vary in time. For this time-varying situation, an on-line adaptive policy is devised to handle this time-varying situation. This on-line policy is based on dynamic lookup of pre-computed off-line optimal policies from a Cached Policy Table (Hwang et al. [1997] and Chung et al. [2002]). The key into this cache table is the parameter set that describes the channel conditions (packet error rate) and the server status (rejection probability for RPRs.) The value is the optimal policy that should be put to practice. Although the optimality of the on-line policy cannot be guaranteed (because of the client-side error and/or latency in determining the channel and

server parameters), it has proven to be a satisfactory solution in a varying environment, especially if the dynamics of the network change are not too fast (cf. the results.)

5.1 Off-line optimal policy

The goal is to find an optimal policy for minimizing the energy consumed by the client based on the characteristics of the client, the wireless channel, and the server. To account for QoS requirements of real applications, the optimal policy is obtained subject to hard constraints on the expected task service time and task loss rate. A task is lost in (or dropped by) any of the client queues (SQ, CQ or PQ) if the queue is full when the task arrives. We formulate the policy optimization problem as a linear program as described next.

Let *x* represent the state of the whole power-managed system and a_x denote an action enabled in state *x.* The constrained energy optimization problem is formulated as a linear objective function with constraints as follows:

$$\mathrm{Min}_{\{f_x^{a_x}\}}\left(\sum_x\sum_{a_x} f_x^{a_x}\gamma_x^{a_x}\right) \tag{5-1}$$

where $f_x^{a_x}$ is the frequency that state *x* is entered in and action a_x is chosen in that state; $\gamma_x^{a_x}$ is the expected cost, which represents the expected energy consumed when the system is in state x and action a_x is chosen. It is calculated as:

$$\gamma_x^{a_x} = \tau_x^{a_x} pow(x, a_x) + \sum_{x'\neq x} p_{x,x'}^{a_x} ene(x, x'), \tag{5-2}$$

where $\tau_x^{a_x} = 1\Big/\sum_{x'\neq x}\sigma_{x,x'}^{a_x}$ denotes the expected duration of time that the system will stay in state x when action a_x is chosen; and $p_{x,x'}^{a_x} = \sigma_{x,x'}^{a_x}\Big/\sum_{x''\neq x}\sigma_{x,x''}^{a_x}$ denotes the probability that the next system state is *x'* when the system is in state x and action a_x is chosen.

This optimization problem is subject to the following conditions.

1) Non-negativity Constraint

$$f_x^{a_x} \geq 0 \tag{5-3}$$

The inequality is implicit in the definition of variable $f_x^{a_x}$. This is because $f_x^{a_x}$ is a frequency, which can only take a nonnegative value.

2) Normalization Constraint

$$\sum_{x}\sum_{a_x} f_x^{a_x}\tau_x^{a_x} = 1 \tag{5-4}$$

This equation sets the summation of all *state-action probabilities* equal to one, which follows from the definition of a probability space.

3) Transfer-Balance Constraint

$$\sum_{a_x} f_x^{a_x} - \sum_{x'\neq x}\sum_{a_{x'}} f_{x'}^{a_{x'}} p_{x',x}^{a_{x'}} = 0, \qquad \forall x \in X \tag{5-5}$$

It is known that if a Markovian process is stationary, then the input rate of each state will be equal to the output rate of that state.

4) Loss Rate Constraint

$$\sum_{x}\sum_{a_x} f_x^{a_x}\tau_x^{a_x}(\delta(\text{SQ full}) + \delta(\text{CQ full}) + \delta(\text{PQ full})) \le T_h \tag{5-6}$$

where $\delta(x) = \begin{cases} 1, & \textit{if } x \textit{ is true;} \\ 0, & \textit{otherwise.} \end{cases}$.

This inequality ensures that the probability that the queue becomes full is less than a preset threshold. This is our way of controlling the request loss rate in the system.

5) Delay Constraint

This constraint limits the average service delay of locally generated SRs, which may be processed locally or remotely. Let $c_x \ge 1$ denote value of the slack factor c used in determining the RTB of an RPR (c.f. Section 4.3) when the system is in state x. With regard to the defined system model, c_x value affects only if the CP is in state CC (Conference) in the global system state x. In the following, this condition is expressed by formula $S_{cp,x}$=CC, where the only available action a_x is "do negotiation".

Let $f_x^{CM|CC}$ denote the frequency of a transition whereby the CP enters state CM from a system state x where the CP was in state CC. It is also the frequency that an RPR is accepted in state x. From Little's theorem (Dshalalow [1997]), for a stationary queueing system, the expected number of service requests waiting in the system is equal to the product of the average incoming rate of the requests and the expected delay experienced by a request. Thus, by viewing the client-server system as a black-box, which is the view typically taken by the client, the delay constraint may be expressed as

$$\sum_{x}\sum_{a_x} f_x^{a_x}\tau_x^{a_x}(q_{s,x} + q_{p,x} + q_{c,x}) + \sum_{x}(T_w + \frac{k}{\mu_c}) f_x^{CM|CC} \le \lambda_c D\,, \tag{5-7}$$

and

$$f_x^{CM|CC} = \sum_{a_x} \sum_{x'} f_x^{a_x} p_{x,x'}^{a_x} \delta(s_{cp,x} = \text{CC} \,\&\, s_{cp,x'} = \text{CM}), \forall x \in X \,. \tag{5-8}$$

where λ_c is the average rate of incoming tasks for the client; $q_{s,x}$, $q_{c,x}$ and $q_{p,x}$ represent the numbers of waiting tasks in the queue SQ, CQ and PQ in state *x*, respectively.

This constraint is explained as follows. The first term on the left-hand side of this inequality calculates the expected total number of SRs waiting in the mobile client; the second term computes the same for the remotely executed SRs (RPRs) on the server, which is explained as follows. Recalling the definition of RTB of an RPR and noting that $f_x^{CM|CC} / \sum_{x'} f_{x'}^{CM|CC}$ is the probability of an accepted RPR in state *x*, it follows that $\sum_x f_x^{CM|CC} (T_w + \frac{k}{\mu_c}) \Big/ \sum_{x'} f_{x'}^{CM|CC}$ limits the average service delay of an accepted RPR on the server. The total rate of SRs migrated to the server equals $\sum_x f_x^{CM|CC}$. According to Little's theorem, the product of these two terms, which is $\sum_x f_x^{CM|CC} (T_w + \frac{k}{\mu_c})$, gives the expected number of RPRs to be executed on the server. Thus, the summation on the left-hand side is the total number of SRs waiting in the black-box system. Since λ_c is the average rate of service requests of the mobile client, again from Little's theorem, this inequality limits the expected service delay seen by an incoming service request to *D*.

Inequality (5-7) is not a linear constraint since c_x, which determines the value of T_w, and $f_x^{CM|CC}$ are both unknown variables. In the following, we discuss how to convert this constraint to a linear constraint. Let $f_x^{CI|CC}$ be the frequency of a transition whereby the CP enters state CI from a system state *x* where the CP was in state CC. It is also the frequency that an RPR is rejected in state *x*. Let $p_{x,im}^{a_x}$ denote the probability that from state *x* where $S_{cp,x}$=CC, the system transits to state *x'*, where $S_{cp,x}$=CI or $S_{cp,x}$=CM. $p_{x,im}^{a_x}$ is constant regardless of the value of c_x. So we have the following linear constraint

$$f_x^{CM|CC} + f_x^{CI|CC} = \sum_{a_x} f_x^{a_x} p_{x,im}^{a_x} \delta(S_{cp,x} = \text{CC}) \,, \ \forall x \in X \,, \tag{5-9}$$

with $f_x^{CM|CC} \geq 0, f_x^{CI|CC} \geq 0$. (5-10)

From equation (4-1), we have

$$\frac{f_x^{CI|CC}}{f_x^{CM|CC}+f_x^{CI|CC}}=\frac{\lambda_s}{\mu_s}\cdot\frac{\mu_c}{(\mu_s-\lambda_s)k(c-1)+\mu_c}\quad\Rightarrow\quad c_x=\frac{k\mu_s-\mu_c}{k\mu_s}+\frac{\rho\mu_c}{k\mu_s(1-\rho)}\frac{f_x^{CM|CC}}{f_x^{CI|CC}}$$

where $\rho=\lambda_s/\mu_s<1$. (5-11)

By substituting (5-11) into (4-2), we obtain

$$T_w=\frac{\rho}{\mu_s(1-\rho)}-\frac{f_x^{CI|CC}}{\mu_s f_x^{CM|CC}}, \tag{5-12}$$

Since for a meaningful RPR timing bound, it is required that $T_w\geq 0$, from (5-12), we obtain

$$f_x^{CI|CC}\leq\frac{\rho}{1-\rho}f_x^{CM|CC} \tag{5-13}$$

After substituting (5-12) into (5-7), we get

$$\sum_x\sum_{a_x}f_x^{a_x}\tau_x^{a_x}(q_{s,x}+q_{p,x}+q_{c,x})+\sum_x\frac{1}{\mu_s}[(\frac{\rho}{1-\rho}+\frac{k\mu_s}{\mu_c})f_x^{CM|CC}-f_x^{CI|CC}]\leq\lambda_c D\,. \tag{5-14}$$

The optimization problem is a linear program (LP) having a linear objective function (5-1) confined by constraints (5-3) to (5-6), (5-9), (5-10), (5-13) and (5-14). It is solved over variables $f_x^{a_x}$, $f_x^{CM|CC}$, $f_x^{CI|CC}$.

5.2 On-line policy

For the on-line policy, we assume that the status of the wireless channel and the aggregated incoming task rate to the server is not *a priori* known. Our solution is to construct a cache table of M×N entries off-line and then employ the table at runtime. Each entry (i,j) in this table corresponds to an optimal DPM policy computed based on the method proposed in Section 5.1 under the condition that the packet error rate of the wireless channel is PER_i and the average incoming task rate to the server is $\lambda_{s,j}$. The indices of the cache table are arranged in an increasing order, i.e., $PER_i < PER_{i+1}$ and $\lambda_{s,j} < \lambda_{s,j}$. The sets $\{PER_i\}$ and $\{\lambda_{s,j}\}$ for which an optimal policy is pre-computed and stored in the table are determined by monitoring the channel and the workload status of the server during a characterization phase and recording the most common sets of conditions. Note that if the pair of online parameters, *PER* and λ_s, is different from any that is stored in the lookup table, then the policy corresponding to the nearest recorded pair of parameter values is chosen.

In contrast to the off-line optimal policy, if during a predetermined period there are no RPRs, the on-line policy will arbitrarily select a task as a REC and send a corresponding RPR to the server. This is needed in order for the client to learn about the condition of the wireless channel and the status of the server.

The client uses profiling and regression to estimate the value of *PER* and λ_s online as is detailed next. Let $APER^{(n)}$ denote the percentage of corrupted packets during the *n*th conferencing session with the server. The predicted value of the packet error rate $PER^{(n)}$ is calculated as:

$$PER^{(n)} = \alpha \cdot APER^{(n)} + (1-\alpha) \cdot PER^{(n-1)} .$$

where α is a coefficient and $0 \le \alpha \le 1$. α should be set to a value closer to one in a fast-changing wireless channel and to a value closer to zero in a slow-changing wireless channel.

Let $\lambda_s^{a,(n)}$ denote the incoming task rate and the average task service time measured on the server side within an n^{th} sliding window. Thus, the predicted incoming task rate for the next sliding window is

$$\lambda_s^{(n)} = \beta \cdot \lambda_s^{a,(n)} + (1-\beta) \cdot \lambda_s^{(n-1)} .$$

where $0 \le \beta \le 1$ is a coefficient. β should be set to a values closer to one if the workload status of the server changes rapidly; otherwise, it should be set closer to zero.

If one of the two conditions takes place:

$(PER_{i-1}+PER_i)/2 < PER^{(n)} \le (PER_i+PER_{i+1})/2 < PER^{(n-1)}$ or

$PER^{(n-1)} \le (PER_{i-1}+PER_i)/2 < PER^{(n)} < (PER_i+PER_{i+1})/2$,

then the policy corresponding to the entry $(i, \cdot)$ will be activated. Similarly, if condition

$(\lambda_{s,j-1}+\lambda_{s,j})/2 < \lambda_s^{(n)} \le (\lambda_{s,j}+\lambda_{s,j+1})/2 < \lambda_s^{(n-1)}$ or

$\lambda_s^{(n-1)} \le (\lambda_{s,j-1}+\lambda_{s,j})/2 < \lambda_s^{(n)} < (\lambda_{s,j}+\lambda_{s,j+1})/2$,

is satisfied, the policy corresponding to entry $(\cdot, j)$ will be activated. Note that "·" represents the unchanged index or index changed based on other conditions. The index *i* and *j* are calculated independently.

The flow diagram of the on-line policy is shown in Figure 8.

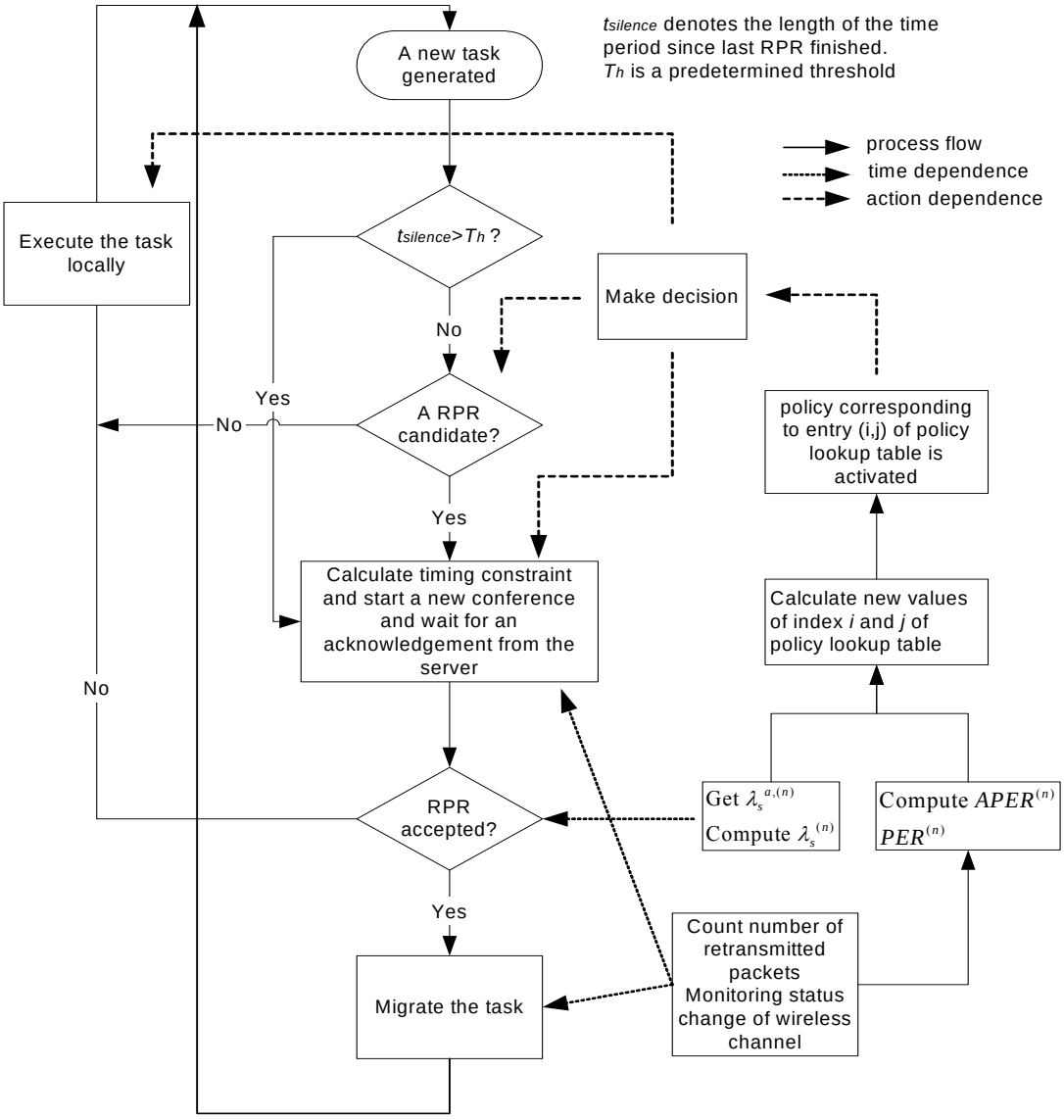

Figure 8. Flow diagram of the on-line policy.

5.3 On-line policy extension to G/G/1 server

In the previous two subsections, we assumed the server with aggregated input service requests can be modeled as an M/M/1 queue. Here we extend the proposed approach to solve the policy optimization problem with a G/G/1 server where the request interval time and service time of the aggregated input service traffic take general distributions

which are unknown to the mobile client. Thus it is not possible to find analytical relationships between P_{reject}, T_w and c, such as those given in equations (4-1) and (4-2), when constructing the policy table.

We continue to assume that the server executes the same scheme as described in section 4.3 for the RPR acceptance and execution. In this case, we can use a two-dimensional policy table; However, this time, each entry (i,j) in the table corresponds to an optimal DPM policy computed under the condition that the packet error rate of the wireless channel and the frequency of RPR rejections over a moving window measurement are PER_i and $P_{reject,j}$, respectively. A predetermined c value is used to determine the RTB of an RPR. Recalling the definition of RTB, ck/μ_c is an upper bound on the average time of an accepted RPR staying on the server. So the delay constraint in section 5.1 will be cast as

$$\sum_x \sum_{a_x} f_x^{a_x} \tau_x^{a_x} (q_{s,x} + q_{p,x} + q_{c,x}) + \frac{ck}{\mu_c} \sum_x f_x^{CM|CC} \le \lambda_c D \tag{5-15}$$

With known values of PER and P_{reject}, the state transition probabilities $p_{x,x'}^{a_x}$ may be obtained for any x, x' and a_x (c.f. definition of $p_{x,x'}^{a_x}$ in equation (5-2)). Thus, by substituting equation (5-8) into (5-15), we obtain a linear constraint

$$\begin{aligned} &\sum_x \sum_{a_x} f_x^{a_x} \tau_x^{a_x} (q_{s,x} + q_{p,x} + q_{c,x}) \\ &+ \frac{ck}{\mu_c} \sum_x \sum_{a_x} \sum_{x' \ne x} f_x^{a_x} p_{x,x'}^{a_x} \delta(s_{cp,x} = \text{CC} \,\&\, s_{cp,x'} = \text{CM}) \le \lambda_c D \end{aligned} \tag{5-16}$$

The policy for each table entry is obtained by solving as a linear program with constraints (5-3) to (5-6) and (5-16). Let RR_N denote the rejection ratio of the last N RPRs. The predicted server rejection probability $P_{reject}^{(n)}$ is:

$$P_{reject}^{(n)} = \beta \cdot RR_N + (1-\beta) \cdot P_{reject}^{(n-1)} \ .$$

where $0 \le \beta \le 1$ is a coefficient. Since RR_N is the latest observed rejection ratio while $P_{reject}^{(n)}$ is an age-weighted average ideally capturing the long-term task rejection ratio, β should be set to a value close to one if the workload status of the server changes rapidly; otherwise it should be set close to zero.

The flow diagram for this policy is similar to that presented in Figure 8, except that different indexing parameters are used in the policy table look-up.

6. EXPERIMENTAL RESULTS

We used a StrongARM SA-1110 processor as the SP in the mobile host. The StrongARM processor was running at a clock frequency of 206MHz. The CP in the host was Orinoco WLAN PC card. The power consumption and state transition times of the StrongARM processor and the Orinoco WLAN PC card are reported in Table 1.

TABLE 1. FEATURES OF STRONGARM SA1110 AND ORINOCO WLAN.

StrongARM SA1110	Busy	Wait	Sleep
Power (mW)	600 (with MEM)	100	0.2
Transition Time	Wait to Busy Busy to Wait	10 us	
	Sleep to Busy	160 ms	
	Busy, Wait to Sleep	90 us	
Orinoco WLAN card	Transmit	Receive	Sleep
Power (mW)	1400	900	50
Transition Time	Wake-up time	34 ms	
	Sleep-down time	62 ms	

In the simulations, we assumed that the average task execution time on the mobile host is 400ms, the conference time is 40ms, and the average RPR data migration time plus the RES pick up time is 80ms. The task incoming rate is 0.625 per second. The Maximum task loss rate is 0.1%. The average task delay constraint is less than 0.8s. We compare the results of our offline and online policies with two baseline policies. These two baseline methods are:

LEO (Local Execution Only) policy: No RPR. The client will execute every task locally.

REF (Remote Execution First) policy: Always try RPR first. For every incoming task, the client will first send an RPR to the server. The client will execute the task locally only if the server rejects the RPR.

Off-line policy

In Figure 9, assume that the state of the wireless channel and the server are unchanged. We will refer to our **Markovian Decision process-based Remote Processing policy** as the **MDRP** policy.

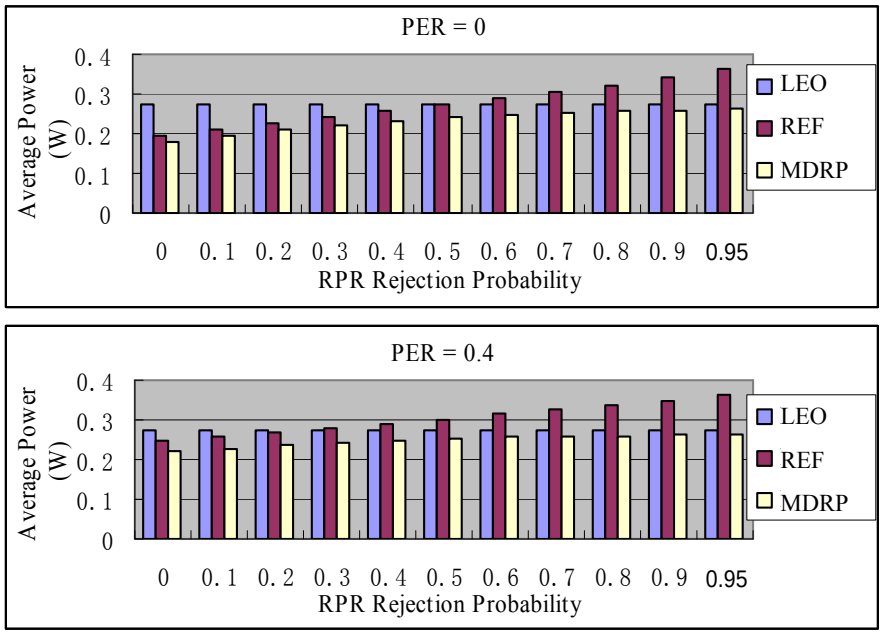

Figure 9. Comparison of simulation results of the three policies with an invariable wireless channel and server.

When the PER and RPR rejection probabilities are small, the REF policy results in large power savings compared with the LEO policy. However, as the PER and RPR rejection probabilities increase, the average power consumption of the REF policy increases and finally significantly outweighs that of the LEO policy. This trend exists because of both the energy wasted by the CP during the RPR negotiations and the extra energy consumed by the SP arising from the more stringent timing constraints (since some time has been wasted for RPR negotiations.) The results demonstrate that the MDRP policy always consumes the least power and achieves power savings as high as 35%.

Next we consider a wireless channel and a server with time-varying characteristics. In this simulation, the server is simulated as an infinite queue with a Markovian process-based task generator (task incoming rates are $\lambda_{s,1}$ and $\lambda_{s,2}$). We assumed that the average task execution time on the server is 40ms and the processing speed of the server is 10 times faster than the client. The remaining model parameters are reported in Table 2. Results of the off-line policy are compared with the two baseline polices in Table 3. In this comparison, the REF policy uses a fixed slack factor *c* which guarantees that, under the worst conditions, the average delay of remotely executed tasks is bounded by the

expected delay *D*. The results demonstrate that the MDRP policy achieves more than 17% power savings compared to both baseline policies. This power saving comes from the fact that the MDRP policy dynamically adjusts the probability of RPR generation and the RTB for remotely executed tasks based on the state of the wireless channel, the server, as well as the mobile client.

TABLE 2. MODEL PARAMETERS OF WIRELESS CHANNEL AND SERVER.

PER1	PER2	$v(1,2)$	$v(2,1)$
0%	20%	1/15000	1/10000
$\lambda_{s,1}$	$\lambda_{s,2}$	$\eta(1,2)$	$\eta(2,1)$
16 per sec.	24 per sec.	1/20000	1/20000

TABLE 3. SIMULATION RESULTS OF THE OFF-LINE POLICY.

Policy	LEO	REF	MDRP
Average Power (W)	0.2742	0.2788	0.2292
MDRP Improvement	17.4%	17.8%	--

On-line policy

In the next two simulations, the server is simulated as an infinite queue with a randomly generated task trace that follows a Markovian process in **Sim1** or a Pareto process in **Sim2**. The parameters of the wireless channel is slowly and randomly increased or decreased. The on-line policy is based on a 5×5 decision table. Simulation results are shown in Table 4.

TABLE 4. SIMULATION RESULTS OF THE ON-LINE POLICY.

	Mode	LEO	REF	MDRP
Sim1	Average Power (W)	0.2742	0.2597	0.2276
	MDRP Improvement	17.0%	13.4%	--
Sim2	Average Power (W)	0.2742	0.2810	0.2364
	MDRP Improvement	14.8%	16.8%	

7. CONCLUSION

A new mathematical framework for extending the lifetime of a mobile host in a client-server wireless network by using remote processing was proposed. The client-server system was modeled based on the theory of continuous-time Markovian decision processes. The DPM problem was formulated as a policy optimization problem and solved exactly by using a linear programming approach. Based on the off-line optimal policy computation, an on-line adaptive policy was developed and employed in practice. This adaptive policy is further extended to solve the problem with a server where the request interval time and the service time assume general distributions. Experimental results demonstrated the effectiveness of our proposed methods.

REFERENCES

BENINI, L., PALEOLOGO, G., BOGLIOLO, A., AND DE MICHELI, G. 1999. Policy optimization for dynamic power management. *IEEE Trans. Computer-Aided Design*, 813–833.

BOYD S. AND VANDENBERGHE L., 2004. *Convex Optimization*. Cambridge University Press.

CHUNG, E.-Y., BENINI, L., BOGLIOLO, A., LU, Y.-H. AND DE MICHELI, G. 2002. Dynamic power management for nonstationary service requests. *IEEE Transactions on Computers*, 1345-1361.

DSHALALOW, J.H. 1997. *Frontiers in queueing: models and applications in science and engineering*. Boca Raton, Fla.: CRC Press.

ELLIOT, E.O. 1963. *Estimates of error rates for codes on burst-noise channels*. *Bell Syst. Tech.J.* 42, 1977-1997.

FEINBERG, E.A., AND SHWARTZ, A. 2002. *Handbook of Markov decision processes: methods and applications*. Kluwer Academic.

HAGGSTROM, O. 2002. *Finite Markov chains and algorithmic applications*. Cambridge Univ. Press, Cambridge, New York.

HWANG, C.-H., AND WU, A. C.-H. 1997. A predictive system shutdown method for energy saving of event-driven computation. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 28-32.

KREMER, U., HICKS, J., AND REHG, J. 2001. A compilation framework for power and energy management on mobile computers. In *Proceedings of International Workshop on Languages and Compilers for Parallel Computing*, Aug. 2001.

OTHMAN, M., AND HAILES, S., 1998. Power conservation strategy for mobile computers using load sharing. *Mobile Computing and Communications Review* 2(1), 44–50.

QIU, Q., WU, Q. AND PEDRAM, M. 2001. Stochastic modeling of a power-managed system-construction and optimization. *IEEE Transactions on Computer-Aided Design*, 1200-1217.

RUDENKO, A., REIHER, P., POPEK, G., AND KUENNING, G., 1998. Saving portable computer battery power through remote process execution. *Mobile Computing and Communications Review* 2(1), 19–26.

RUDENKO, A., REIHER, P., POPEK G., AND KUENNING G. 1999. The remote processing framework for portable computer power saving. In *Proceedings of ACM Symposium on Applied Computing*, San Antonio, TX, Feb. 1999.

SMAILAGIC, A., AND ETTUS, M. 2002. System design and power optimization for mobile computers. In *Proceedings of VLSI on Annual Symposium, IEEE Computer Society ISVLSI*, 2002, 15 –19.

YANG, H., AND ALOUINI, M.-S. 2002. *A hierarchical Markov model for wireless shadowed fading channels*. In *Proceedings of Vehicular Technology Conference*, 640-644.

ZORZI, M., RAO, R.R., AND MILSTEIN, L.B. 1998. Error statistics in data transmission over fading channels. *IEEE Transactions on Communications* 46(11).

BIOGRAPHIES

Peng Rong received B.S. and M.S. degree in Electronic Engineering from Tsinghua University, China in 1998 and 2001, respectively. He received a Ph.D. degree in Electrical Engineering at the University of Southern California in 2006. He is currently a senior design engineer at Brocade Communications Systems. His research interests are in the area of system-level power management and low-power design.

Massoud Pedram received a B.S. degree in EE from Caltech in 1986 and a Ph.D. degree in EECS from UC-Berkeley in 1991. He joined the department of Electrical Engineering at the University of Southern California in 1991 where he is currently a full professor and director of the computer engineering division. From 1987 to 1989, he worked at the Xerox Palo Alto Research Center. Dr. Pedram has served on the technical program committee of many technical conferences, including the Design Automation Conference, and the Design and Test in Europe Conference. He co-founded and served as the Technical Chair and General Chair of the Int'l Symposium on Low Power Electronics and Design in 1996 and 1997, respectively. He has published four books and more than 350 technical papers. His research has received a number of awards including two ICCD Best Papers, two DAC Best Papers, an IEEE T-VLSI Best Paper, and an IEEE T-CAS-II Best Paper. He is a recipient of the NSF's Young Investigator Award (1994) and the Presidential Faculty Fellows Award (1996). Dr. Pedram, who is a Fellow of the IEEE and an ACM Distinguished Scientist, currently serves as the Editor-in-Chief of the ACM Transactions on Design Automation of Electronic Systems. His current work focuses on low power electronics, energy-efficient information processing systems, and green computing.