Scheduling Algorithms for Downlink Services in Wireless Networks: A Markov Decision Process Approach

William A. Massey
ORFE Department
Engineering Quadrangle, Princeton University
Princeton, NJ 08544

K. G. Ramakrishnan, M. Aravamudan and G. Pai
Motorola Inc.
3 Highwood Drive
Tewksbury, MA 01876

***Abstract* - We model the scheduling of emerging wireless data services as a Markov decision process. These services are characterized by real-time downlink data transfers such as those needed for personalized traffic, weather and business updates. Characterizing each differing service by its own time-criticality as well as reward and penalty values, we can numerically compute an optimal Markov decision scheduling rule for serving these requests. These computations suggest that we can formulate a simple, near optimal rule for scheduling these services efficiently.**

***Keywords*: Markov Decision Processes, 3G-Wireless, Cognitive Radio, Optimal Scheduling.**

I. Introduction

This paper proposes both a mathematical model and a scheduling algorithm for emerging new services in current day wireless networks. These services can be called personalized, push technology services. A typical example will illustrate our model. Imagine an application server for a regional wireless network that knows the current traffic flows in the major highways. A customer can subscribe to this traffic update service, by specifying a personalized profile–call this *myTraffic*–of the congestion levels for the highways that the customer is interested in knowing. When starting the morning commute in the car, the customer initiates a request to know the current traffic flows. The application server gets the request, processes it according to a scheduling algorithm, and downloads the information to the customer's mobile device. The application server needs to service this request quickly, as the traffic patterns are time sensitive. If the application server is unable to service this job within a certain time, then the job is aborted and a penalty value accrues. If the application server is able to honor the request, then a revenue value accrues. Thus the problem parameters are:

1. The arrival rate of such requests.
2. The time-criticality of the response.
3. The revenues (penalties) for servicing (aborting) requests.
4. The amount of CPU time that the request takes to be processed. This depends on the nature of the customer's personal profile, and added features of the service, such a suggesting alternate routes.

In modelling this problem, we are not concerned with the system by which the server updates its knowledge base. This is an interesting problem of data acquisition, but not the subject of this paper. We assume that the application server always has an up-to-date knowledge base. The system is stochastic because of the random nature of arrivals and varying service time requirements. If the application server is processing only one type of request, say *myTraffic*, then one can easily devise an optimal scheduling policy, which is just to do *first-come-first-served* (FCFS). But if there are multiple such services (*myBusiness*, *myWeather*, etc.), each with its own reward/penalty/time-criticality characterizations, then the optimal policy is not immediately obvious. The problem that we are addressing in this paper is to find an optimal scheduling policy that maximizes the expected revenue for the wireless service provider, given a multiplicity of services.

It should be emphasized that our scheduling algorithm works at the application level. This scheduling has to be consistent with the scheduling and rate allocation at the link level. For this consistency reason, our model has to interact with the protocols like HSDPA and 1x-Ev-Dv (see [4] and [5]) in a collaborative manner. We do not address this interaction in this paper.

The services we consider are all "downlink" services; i.e., the network has the knowledge base, and pushes the data down to the customer's end device. The push can be initiated by the customer or automated to happen when certain other events happen. There are interesting applications in the "uplink" services, where the data is uploaded from the end device to the network. A typical example of an uplink service is a "picture upload" service. All current day mobile phones are equipped with digital cameras. After the picture is taken, a customer can request an upload of the picture to the network. Based on the completion time and the size of the upload, revenue accrues to the service provider. The scheduling problem is to schedule the picture uploads so as to maximize the accrued revenue. We do not address "uplink" services in this paper.

The above services are examples of "cognitive radio" services, which was introduced by J. Mitola [6] in 1999. In that Ph.D. dissertation, Mitola describes a detailed software architecture and language to describe such service applications.

Markov decision processes (MDP) are a special class of dynamic programming models, which were originally proposed by Bellman [3] to solve a deterministic n-step optimization problem using a recursive algorithm. MDP is dynamic programming under uncertainty, where the actual transitions and their rewards are probabilistic. A recursion, similar to deterministic dynamic programming, is developed to solve for the expected maximum or minimum. MDP has been extensively used in developing admission control policies in computer and communication systems, see Mitra, Reiman, and Wang [7] as well as examples in Ross [8].

The MDP-based optimization approach, while mathematically elegant, poses a formidable computational challenge. One has to walk through the state space of the system being modelled to evaluate the optimal solution. For real-time systems, we need a quick and easy way to make a decision transition, based only on the current state of the system. This is possible only if there is a closed-form solution to the MDP system being analyzed. In a seminal result on MDP, the $c\mu$-rule (see Bertsekas [1]) is an optimal closed-form solution to a queueing system. The $c\mu$-rule is simple to state and easy to implement. The optimal rule states that "Among the currently waiting customers to be served, pick the one with the largest product of *revenue* and its *service rate*." One of the first references to this rule can be found in Cox and Smith [2] in the context of optimizing a priority queue.

We propose a similar rule of scheduling for our application server which can be implemented in real time. Unlike the system analyzed in the $c\mu$-rule, our application is more complicated, and the $c\mu$-rule does not directly apply. The complication is due to the penalty for abandonment of a customer because of the waiting time exceeding a certain value. This reward for service but penalty for abandonment paradigm makes the MDP harder to analyze. However, as described in the computational results section, many computational results on various systems point to a conjecture that is similar to the $c\mu$-rule, in the interior of the state space. On the boundaries, a threshold policy seems to be optimal for a restricted set of cases.

The outcome of our research is a scheduling policy for downlink scheduling, that can be stated as follows:

1. Statically order the classes in ascending order of the metric, $r\mu + \beta s$, where r is the revenue for serving the job in a class, μ is the average service rate, s is the magnitude of the penalty for abandonment (a positive number), and β is the abandonment rate.

2. When a server becomes free, and all job queues are non-empty, serve the job with the largest class index (largest $r\mu + \beta s$).

The paper is organized as follows. Section II describes the MDP system and develops the recursion formulas. Section III describes the computational results. Section IV proposes the optimal policy conjectures as stated above, and section V offers concluding remarks.

II. Markov Decision Model

To specify the MDP system, we first define $\{\, \mathbf{X}(t) \mid t \geq 0 \,\}$ to be a continuous time Markov chain

$$X(t) = (Q_1(t), C_1(t), \ldots, Q_K(t), C_K(t))\,, \tag{1}$$

where $Q_i(t)$ equals the number of class i customers that are waiting to be served (i.e. in the queue) at time t and $C_i(t)$ equals the number of class i customers that are being served at time t. The range for the index i is $1, 2, \ldots, K$ where K equals the total number service classes.

If S equals the set of *states* for this process, then each $\sigma \in S$ is of the form

$$\sigma = (q_1, c_1, \ldots, q_K, c_K), \tag{2}$$

where the q_i and c_i are non-negative integers. Moreover, these states satisfy the following constraint:

$$\sum_{i=1}^{K} c_i \leq c_{max}. \tag{3}$$

This simply means that our queueing system has no more than c_{max} servers.

Now let $\mathit{\Pi}$ be the set of *partial transitions* which correspond to events that happen outside the control of the application server. These transitions are followed instantaneously by a *decision transition* which is in the control of the application server. Together, they form a single step transition for this Markov process. The partial transitions can be described as follows:

1. Arrival of a class i customer:

   $$q_i \longrightarrow q_i + 1. \tag{4}$$

   The time between such arrival partial transitions are exponentially distributed with rate λ_i.

2. Abandonment by a class i customer:

   $$q_i \longrightarrow q_i - 1 \tag{5}$$

   We define the abandonment time, that a class i customer waits for service, to be exponentially distributed with rate β_i. Since we assume that these customers act independently of each other, then the times between abandonment partial transitions are exponentially distributed with rate $q_i\beta_i$. Note that this automatically guarantees that no such transition will occur when $q_i = 0$.

3. Service completion for a class i customer:

$$c_i \longrightarrow c_i - 1. \tag{6}$$

We define the time that a class i customer spends in service is exponentially distributed with rate μ_i. Since we assume that these customers act independently of each other, then the time between such service partial transitions are exponentially distributed with rate $c_i\mu_i$. This automatically guarantees that no such transition will occur when $c_i = 0$.

Consider the two class case $K = 2$. The above transitions are partial ones for the following reasons. It is appropriate to assume that no decision affects the rate of new customers arriving to acquire a specific service. Customers in queue may tire of waiting for service and then choose to abandon the system solely on the basis of how long their wait has been. Finally, we assume that customers in service will complete their task independently of any decision made.

Given a state σ, the time until some partial transition α occurs is exponentially distributed with rate $\sum_{i\in K} \lambda_i + q_i\beta_i + c_i\mu_i$. Let $p_\sigma(\alpha)$ equal the probability that this partial transition equals α. If α is the partial transition for a class i customer arrival, then

$$p_\sigma(\alpha) = \frac{\lambda_i}{\sum_{j\in K}(\lambda_j + \beta_j q_j + \mu_j c_j)}. \tag{7}$$

If α is the partial transition for a class i customer abandonment, then

$$p_\sigma(\alpha) = \frac{q_i\beta_i}{\sum_{j\in K}(\lambda_j + \beta_j q_j + \mu_j c_j)}. \tag{8}$$

Finally, if α is the partial transition for a class i customer service completion, then

$$p_\sigma(\alpha) = \frac{c_i\mu_i}{\sum_{j\in K}(\lambda_j + \beta_j q_j + \mu_j c_j)}. \tag{9}$$

To describe the decision transitions, we let $\Delta(\sigma,\alpha)$ equal the set of *decision states* for the current state σ and partial transition α, where $\Delta(\sigma,\alpha) \subseteq S$. These partial transitions become complete single step transitions for our Markov process once we specify the corresponding decision transition. The general nature of these decision transitions is to determine which current customer in the queue should now enter service (if a server is free).

Let $r_\sigma(\alpha)$ be the *general reward* at state σ where the next partial transition is α. For our model, $r_\sigma(\alpha) = 0$ if α is any arrival partial transition and $r_\sigma(\alpha) = -s_i$, where $s_i > 0$, is a penalty if α is an abandonment partial transition for a class i customer with $q_i > 0$. However, $r_\sigma(\alpha) = r_i > 0$ is a reward if α is a service completion partial transition for a class i customer with $c_i > 0$.

Given n transition steps for our Markov process, we define our *maximal value function* $v_\sigma(n)$ to be the sum of the rewards (or penalties) made at the n sequential, partial transition steps, following the initial state σ, that achieves the maximum expected revenue.

To make these transitions, we need to couple each partial transition with a decision one to select the next state. We define $\delta(\sigma,\alpha,n) \in \Delta(\sigma,\alpha)$ to be the *transition decision rule given n remaining steps* following a partial transition step that is α and the state before these transitions is σ. We now make the following modelling assumptions for our decision transitions:

- Partial transitions do not depend on the past history or the next set of decision states.
- Rewards for partial transitions do not depend on the past history or the next set of decision states.
- New decision transitions are based only on the current state and transition.
- New decision transitions are only influenced by the current state, the current partial transition, and the number of remaining transitions.

The decision transition gives us our new transition state. Given a partial transition α occurring at state σ, we "decide" to select a new state τ with probability $p^\delta_\sigma(\tau|\alpha)$. For a given policy, these decisions are deterministic and so with probability one, we have $\tau = \delta(\sigma,\alpha,n)$ if there are n-transition steps remaining. If $p^\delta_{\sigma\tau}$ equals the one step transition probability that takes us from state σ to state τ, then we have

$$p^\delta_{\sigma\tau} = \sum_{\alpha\in\Pi} p^\delta_\sigma(\tau|\alpha)p_\sigma(\alpha). \tag{10}$$

Now we define $v_\sigma(n)$ to be the maximal value function over all policies as described above by the recursion relation

$$\begin{aligned} v_\sigma(0) &= 0 \\ v_\sigma(n+1) &= \sum_{\alpha\in\Pi} p_\sigma(\alpha)\cdot\left(r_\sigma(\alpha) + \max_{\tau\in\Delta(\sigma,\alpha)} v_\tau(n)\right). \end{aligned}$$

Note that this MDP formulation differs from similar recursion equations found in Ross [8]. This is due to the fact that we have a *partial transition* that is decision independent and is occuring before the decision transition, which corresponds to the probability $p_\sigma(\alpha)$. On the other hand, it is these partial transitions that dictate the next immediate reward.

The optimal decision rule simply defines $\delta(\sigma,\alpha,n)$ to be that state that satisfies the equation

$$v_{\delta(\sigma,\alpha,n)}(n) = \max_{\tau\in\Delta(\sigma,\alpha)} v_\tau(n).$$

III. Numerical Results

We now implement the recursion algorithm and conduct numerical experiments with different classes and randomly generated parameters for these classes. For brevity, we summarize these results using two examples whose parameters

Table 1: Problem parameters for the 2-class problem. Number of servers = 10.

Class ID	Arrival Rate λ jobs/sec	Service Rate μ jobs/sec	Aband. Rate β jobs/sec	Revn. r \$/job	Penl. s \$/job
1	94.0	30.0	4.0	55.0	291.0
2	96.0	158.0	88.0	38.0	90.0

Table 2: Problem parameters for the 3-class problem. Number of servers = 10.

Class ID	Arrival Rate λ jobs/sec	Service Rate μ jobs/sec	Aband. Rate β jobs/sec	Revn. r \$/job	Penl. s \$/job
1	45.0	3.0	18.0	8.0	5.0
2	52.0	36.0	29.0	1127.0	23.0
3	22.0	97.0	43.0	1191.0	144.0

are given in Tables 1 and 2. Both examples have 10 servers available for serving jobs. We now partition the state space of the system into two parts: the strict interior and the boundary. The boundary corresponds to states where some classes of customers are absent from at least one of the queues.

A. Results for the Interior State Space for the 2-Class Problem

The interior space is characterized by non-empty queues where either all servers are busy or one or more servers is free. Table 3 shows the optimal policy decisions made for the 2-class problem, a typical interior state, and increasing time horizons. The state is (10, 8, 10, 1). When the partial transition is listed in the left column, we describe the corresponding decision transitions in every other column for that same row. We use the convention of having A^{++} denote the addition of a single customer to either the number of class i customers in queue or in service. Similarly, A^{--} denotes the deletion of a single customer from either the number of class i customers in queue or in service. When there is decision to make no changes to the state, we either do not list the corresponding partial transition or we denote this as a null event $\emptyset$.

It is apparent that the optimal policy is to give the server to class 2 and always queue a class 1 job whenever there is a free server and there is a waiting or arriving class 2 job. Identical results were obtained for other interior states. Time horizons larger than 20 steps were not studied because of the prohibitive computational burden.

B. Results for the Boundary States of the 2-Class Problem

Table 4 shows the optimal policies when the class 2 queue is empty. The optimal policy seems to be that the free servers

Table 3: Optimal decisions in the interior states for the 2-class problem.

Partial Transition	Optimal Decisions for State $(10, 8, 10, 1)$			
	Time Horizons			
	5	10	15	20
Any Arrival or Any Service Completion	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}

Table 4: Optimal decisions in the boundary states for the 2-class problem.

Partial Transition	Optimal Decisions for State $(Q_1, 8, 0, 1)$				
	Q_1 Values				
	0	10	20	40	60
Class 1 Arrival	$\emptyset$	$\emptyset$	$\emptyset$	Q_1^{--} C_1^{++}	Q_1^{--} C_1^{++}
Class 2 Arrival	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}	Q_2^{--} C_2^{++}
Any Service Completion	$\emptyset$	$\emptyset$	$\emptyset$	Q_1^{--} C_1^{++}	Q_1^{--} C_1^{++}

are kept in reserve for class 2 arrivals up to a point of class 1 queue build up. When class 1 queue size is beyond 20, then the policy switches to giving the free server to class 1. This threshold policy trades off the penalty of abandonment of class 1 jobs with the potential future revenues of class 2 jobs.

C. Results for the Interior and Boundary State Space of the 3-Class Problem

Table 5 shows the results in the interior and boundary space of the 3-class problem. Results are shown for the interior queueing state $(10, 10, 10)$ as well as the two boundary queueing states $(10, 0, 10)$ and $(10, 10, 0)$. The *queueing state* here is a 3-tuple that indicates the queue sizes and a *service state* is defined in a similar manner. All the queueing states here have the same service state $(4, 3, 3)$. It is again interesting to note that the policy gives the next available server to the job with the largest class index. Note that in the last column, where class 3 has an empty queue, it is class 2 that now gets the priority to be served, since class 2 has the second largest index. Abandonment events are not shown since no decision is made for these events.

IV. Conjectured Optimal Policy

The computational experiments suggest an optimal policy in the interior and boundary states. If we compute the metric $r\mu + \beta s$ for each class, then the job classes are ordered in ascending order of their metric, both for the 2 and 3 class

Table 5: Optimal decisions in the interior and boundary state space for the 3-class problem.

Partial Transition	Optimal Decisions for the Case of $c_1 = 4$, $c_2 = 3$ and $c_3 = 3$		
	Queueing State Space q_1, q_2, q_3		
	$(10, 10, 10)$	$(10, 0, 10)$	$(10, 10, 0)$
Any Service Completion	Q_3^{--} C_3^{++}	Q_3^{--} C_3^{++}	Q_2^{--} C_2^{++}

examples shown above. Table 3 shows the transitions in the interior of the state space (all queues non-empty) and it is clear that the optimal policy is *to place the job with the largest $r\mu + \beta s$ in service when a server becomes free*. Interestingly, this same optimal policy is also exhibited in the boundary states where all but one queue are non-empty, as shown in Table 5 for the 3 class problem. Table 4 however, exhibits a threshold policy in the boundary states which is discussed in Section III.B. The conjectured optimal scheduling policy is:

1. For all interior states, place the job with the highest $r\mu + \beta s$ in service.

2. For all boundary states, where some queues are empty, adopt a threshold policy where the threshold values are to be determined.

It remains to prove this conjecture rigorously in a general framework.

V. Conclusions

We model emerging downlink wireless data services using Markov decision processes and propose an efficient scheduling algorithm that maximizes the expected revenue for the service provider. The model takes into account the revenue, penalty, and time sensitivity of data to be downloaded to the customer's end device. The decision algorithm exhibits an optimal policy in the interior state space that is an extension of the classical $c\mu$-rule to serve the customer class with the largest sum of the product of the revenue and the service rate plus the product of the penalty and the abandonment rate. The optimal policy for the boundary states requires further investigation.

References

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume Two*, Athena Scientific, 1995.

[2] D.R. Cox and W.L. Smith, *Queues*, 1961, London: Methuen & Co. Ltd.

[3] R. Bellman, *Dynamic Programming*, 1957, (Dover Edition, 2003).

[4] HSDPA and 1xEv-Dv Harmonization Opportunities, *3GPP-3GPP2 Joint Meeting on Harmonization of High Speed Data Services*, Nov 13-14, 2001, www.3gpp.org.

[5] J. Korhonen, *Introduction to 3G Mobile Communications-2nd Edition*, Artech House Inc., Norwood, MA, 2003.

[6] J. Mitola, *Cognitive Radio-Model-based Competence for Software Radios*, Licentiate Thesis (Stockholm KTH), September 1999.

[7] D. Mitra, M. I. Reiman and J. Wang, *Robust Admission Control for Heterogeneous ATM Systems with Both Cell and Call QoS Requirements*, in Teletraffic Contributions for the Information Age, Proc. ITC-15, eds. V. Ramaswami, P.E. Wirth, North Holland, Amsterdam, pp. 1421–1432.

[8] S. M. Ross, *Introduction to Stochastic Dynamic Programming*, Academic Press Inc., 1983.