Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Rational Controllers in Computer/Communication Networks

Pierre T. Kabamba [*] **Wen-Chiao Lin** [**] **Semyon M. Meerkov** [*]
Choon Yik Tang [***]

[*] *Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: kabamba@umich.edu, smm@eecs.umich.edu)*
[**] *Sensors & Decision Systems Group, Idaho National Laboratory, Idaho Falls, ID 83415 USA (e-mail: Wen-Chiao.Lin@inl.gov)*
[***] *School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK 73019 USA (e-mail: cytang@ou.edu)*

Abstract: Traditional feedback controllers are used to ensure system operation in a desired regime. In contrast, rational controllers are intended to determine a regime most appropriate for the system, given varying conditions of its operation. In this paper, two types of rational controllers are described and their applications in wireless personal area networks and sensor networks are presented.

Keywords: Adaptation and learning, control of networks, sensor networks, wireless networks, power efficient computing/communications

1. INTRODUCTION

Computer/communication networks consist of devices that may operate in two or more regimes. For instance, Wireless Personal Area Networks (WPANs) typically include a CPU and a disk drive that may operate in several states, each to be selected so that the required performance is achieved while the power is conserved. Similarly, in Sensor Networks (SNs), each sensor may be either active or asleep in order to ensure both the desired level of network performance and minimal battery depletion. In these examples, each device must be equipped with a controller, which selects an appropriate device state in an intelligent manner. We refer to them as *rational controllers* (RCs). This paper is intended to introduce such controllers and describe their application in WPANs and SNs. The main original contributions are demonstrations of RCs' abilities to select optimal operating states of devices in a WPAN and, in SNs, to improve duty cycle fairness among sensors, while achieving the required network performance. The remainder of this paper is organized as follows. Section 2 introduces two types of RCs and addresses issues associated with their design; Sections 3 and 4 present applications to WPANs and SNs, respectively; finally, Section 5 provides the conclusions. Due to space limitations, many details are omitted and can be found in Flinn et al. (2005) and Kabamba et al. (2007a,b,c). For convenience of the reader, a list of abbreviations is given in Appendix A.

2. RATIONAL CONTROLLERS

2.1 General Properties

Unlike traditional controllers, which force a system to operate in a desired regime, RCs are intended to select a regime, which is most appropriate for system operation. To accomplish this, they are endowed with two properties: ergodicity and selectivity. The *ergodicity property* implies that an RC visits each state in its state space (referred to as the *decision space*). The *selectivity property* implies that the residence time in states associated with a *smaller penalty* is longer than that in states with a *larger penalty.* A positive parameter, referred to as the *level of selectivity* (LS), quantifies how much longer the residence time in "good" states is as compared with "bad" ones. Dynamical systems with these properties have been introduced and studied in Tsetlin (1973), Varshavsky (1973), Meerkov (1979), Narendra and Thathachar (1989), and Kabamba et al. (2007b,c). Below, we describe two specific RCs, which are used later in WPAN and SN applications.

2.2 Rational Probabilistic Deciders

We describe here an RC introduced in Kabamba et al. (2007b). First, we define *probabilistic deciders* (PDs). A PD is a stochastic system, defined by a quadruple $(\mathcal{S}, \Phi, N, \mathcal{P})$, where $\mathcal{S} = \{1, 2, \ldots, s\}$ is the decision space, $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_s\}$ is the penalty function, $N \in (0, \infty)$ is the LS, and $\mathcal{P} = \{P_1(\varphi_1; N), \ldots, P_s(\varphi_s; N)\}$ is a set of transition probabilities. Whenever the PD is in state $i \in \mathcal{S}$ at time n, the probability of it making a state transition at time $n + 1$ is $P_i(\varphi_i; N) \in (0, 1)$. Moreover, when a state transition occurs, the PD changes to any other state with equal probability. The dynamics of a PD can be described by an ergodic Markov chain and the steady state probability of the PD being in state $i \in \mathcal{S}$ will be denoted by $\kappa_i(\varphi_i; N)$.

A PD is a *rational probabilistic decider* (RPD) if $\varphi_i < \varphi_j$ implies $\kappa_i(\varphi_i; N) > \kappa_j(\varphi_j; N)$, and, moreover, $\frac{\kappa_i(\varphi_i;N)}{\kappa_j(\varphi_j;N)} \rightarrow$

∞ as $N \to \infty$. Note that as N becomes arbitrarily large, RPDs require an arbitrary long time to converge to steady state.

2.3 Ring RCs

A ring RC, introduced in Meerkov (1979), has a state space, $X = [0, 1)$, and penalty function $\varphi(x) > 1$, $x \in X$. Its dynamics are described by

$$\begin{cases} \dot{y} &= \varphi^N(\{y\}), \\ x &= \{y\}, \end{cases} \tag{1}$$

where x is the state, $\{y\}$ denotes the fractional part of y, and N is a positive integer representing LS. It is possible to show that

$$\frac{T_N(x^1)}{T_N(x^2)} \approx \left(\frac{\varphi(x^2)}{\varphi(x^1)}\right)^N, \tag{2}$$

where $T_N(x^1)$ and $T_N(x^2)$ denote the time during which the ring element is in the vicinity of $x^1 \in X$ and $x^2 \in X$, respectively. This implies that the ring RC spends more time at points where it is less penalized, and, furthermore, as $N \to \infty$, the ring RC almost always resides at the least penalized state. Unlike RPDs, ring RCs may converge to the steady state arbitrarily fast.

2.4 Design Issues

The following issues should be addressed in designing systems controlled by RCs:

First, an appropriate type of RCs should be selected for the application at hand. In particular, the decision space should be defined, and how the RCs make transitions among the states should be specified. This affects the performance characteristics of the system such as the convergence time to steady state.

Second, the penalty functions for the RCs should be chosen properly. Specifically, how the RCs are penalized should be designed based on the application at hand and performance specifications.

Finally, the dynamics of the resulting control system should be analyzed, and the LS should be selected to ensure that the desired behavior is achieved.

Below, this approach is illustrated by applications to WPANs and SNs.

3. RCs IN WIRELESS PERSONAL AREA NETWORKS

In this section, we use RPDs for power management in WPANs. To simplify the problem, consider a WPAN containing only a processor and a disk drive that, together, have to perform computing and read/write tasks. We first describe the models for job arrival, the processor, and the disk drive. Then we introduce an architecture for power management using RCs. Finally, we evaluate the system performance under several job arrival scenarios.

3.1 Modeling

Job arrival: We conider a sequence of jobs, identified by the index k, so that the kth job requires $w_p(k)$ cycles of computing and $w_d(k)$ bytes of reading from/writing to the disk drive. The sequences $\{w_p(k), k = 1, 2, \ldots\}$ and $\{w_d(k), k = 1, 2, \ldots\}$ are modeled as random processes taking positive integer values.

Processor: We assume that the processor is capable of operating at n_p pairs of frequencies and voltages, denoted $(f_1, v_1), (f_2, v_2), \ldots, (f_{n_p}, v_{n_p})$. For each pair (f_i, v_i), let p_i denote the power consumed. We also assume that both frequency and voltage are kept constant during the processing of each job, and that they are allowed to change upon the completion of a job. Thus, we denote the frequency, voltage, and power used during the processing of the kth job as $f(k)$, $v(k)$, and $p(k)$, respectively. Therefore, the energy consumed by the processor to complete the kth job is given by

$$e_p(k) = p(k)\frac{w_p(k)}{f(k)}. \tag{3}$$

Disk drive: The disk drive is assumed to have three modes of operation: *active*, *idle*, and *asleep*. Whenever a read/write job is requested, the disk drive switches to the active mode, reading/writing at a fixed data rate of s bytes per second. Once the job is completed, it switches to the idle mode, stays there for τ seconds, and then switches to the sleep mode. The parameter τ, referred to as the *threshold*, is allowed to take n_d values, denoted $\tau_1, \tau_2, \ldots, \tau_{n_d}$. We also assume that the threshold is kept constant during the processing of each job, and is allowed to change upon the completion of a job. Thus, we denote the threshold used during the processing of the kth job as $\tau(k)$.

Labeling the active, idle, and asleep modes by 1, 2, and 3, respectively, the power/energy consumption of the disk drive is characterized by the matrix

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1 & \Phi_{12} & \Phi_{13} \\ \Phi_{21} & \phi_2 & \Phi_{23} \\ \Phi_{31} & \Phi_{32} & \phi_3 \end{bmatrix}, \tag{4}$$

where $\phi_i > 0$ is the power used in mode i, and $\Phi_{ij} > 0$ is the energy used to switch from mode i to mode j. As it follows from the above, the energy consumed by the disk drive during the processing of the kth job is given by

$$e_d(k) = \begin{cases} \phi_1 \dfrac{w_d(k)}{s} + \Phi_{12} + \phi_2\Big(\dfrac{w_p(k)}{f(k)} - \dfrac{w_d(k)}{s}\Big) + \Phi_{21}, \\ \qquad\qquad \text{if } \dfrac{w_p(k)}{f(k)} - \dfrac{w_d(k)}{s} \le \tau(k), \\ \phi_1 \dfrac{w_d(k)}{s} + \Phi_{12} + \phi_2 \tau(k) + \Phi_{23} + \cdots \\ \cdots + \phi_3\Big(\dfrac{w_p(k)}{f(k)} - \dfrac{w_d(k)}{s} - \tau(k)\Big) + \Phi_{31}, \\ \qquad\qquad \text{if } \dfrac{w_p(k)}{f(k)} - \dfrac{w_d(k)}{s} > \tau(k), \end{cases} \tag{5}$$

where the first alternative in (5) does not trigger the sleep mode, and the second one does.

3.2 Control Architecture

We assume that the processor and disk drive are equipped with RCs. Upon completion of each job, the processor RC selects the pair of frequency and voltage to be used for the next job. Hence, the processor RC determines the sequence $\{(f(k), v(k)), k = 1, 2, \ldots\}$. Similarly, upon completion of each job, the disk drive RC selects the threshold to be used for the next job and, thus, determines the sequence $\{\tau(k), k = 1, 2, \ldots\}$.

These RCs are used in a control architecture illustrated in Fig. 1, where each RC takes its decision based on the energy per cycle consumed by the WPAN as a whole over the just completed job. Specifically, the processor and disk drive RCs select $(f(k), v(k))$ and $\tau(k)$, respectively, based on $\frac{e_p(k-1)}{w_p(k-1)} + \frac{e_d(k-1)}{w_p(k-1)}$.

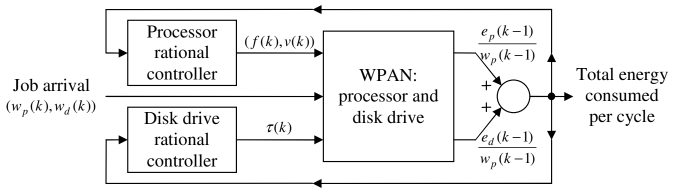

Processor rational controller
$(f(k), v(k))$
Job arrival
$(w_p(k), w_d(k))$
WPAN: processor and disk drive
Disk drive rational controller
$\tau(k)$
$\frac{e_p(k-1)}{w_p(k-1)}$
$\frac{e_d(k-1)}{w_p(k-1)}$
+
+
Total energy consumed per cycle

Fig. 1. Control architecture.

3.3 Rational Controllers: RPDs

As shown in Fig. 1, the RCs have energy per cycle as their inputs and decisions as their outputs. In this application, the RCs are selected as RPDs with n_p or n_d states for the processor or disk drive RC, respectively. The RPD leaves its state, occupied at time $k-1$, with probability $P^N\left(\frac{e(k-1)}{w_p(k-1)}\right)$ or remains in it with probability $1 - P^N\left(\frac{e(k-1)}{w_p(k-1)}\right)$, where $e(k-1) = e_p(k-1) + e_d(k-1)$, N is the LS, and $P : (0, \infty) \to (0, 1)$ is a strictly increasing penalty function; upon leaving its current state, the RPD transits to all other states equiprobably.

The function P and number N are the parameters of the RCs to be selected so that the dynamics and steady-states of the closed-loop system are desirable.

3.4 Scenarios

To evaluate the efficacy of this control architecture, we consider a sequence of $10,000$ jobs and four scenarios for job arrival: (1) *high load*, characterized by $w_p(k) = 10^9$; (2) *medium load*, with $w_p(k) = 5 \times 10^8$; (3) *low load*, with $w_p(k) = 10^8$; and (4) *variable load*, defined as high load for the first $3,334$ jobs, medium load for the next $3,333$ jobs, and low load for the last $3,333$ jobs. For each of these four scenarios, we assume that $w_d(k) = 4 \times 10^5$.

The processor under consideration is the Intel 80200 XScale processor (see Intel Corporation (2005)). Although this processor can operate at seven frequencies, we consider only the lowest and highest frequencies, i.e., 333 MHz and 733 MHz, respectively. For the 333 MHz frequency, the voltage is 1.0 V, whereas for the 733 MHz frequency, the voltage is 1.5 V. As indicated in Pereira et al. (2002), the power consumption at (333 MHz, 1.0 V) is 0.174 W, and at (733 MHz, 1.5 V) is 0.8295 W.

The disk drive considered is the Hitachi 1GB microdrive. This disk drive is capable of reading/writing at a fixed data rate of 4 MB/sec. Although it is capable of operating in four modes, we consider only the active, idle, and asleep modes. For this device, the power/energy consumption matrix $\mathbf{\Phi}$ defined in (4) has the following numerical values (see Flinn et al. (2004)):

$$\mathbf{\Phi} = \begin{bmatrix} 1.18\,\mathrm{W} & 0\,\mathrm{J} & 0.3\,\mathrm{J} \\ 0.09\,\mathrm{J} & 0.76\,\mathrm{W} & 0.3\,\mathrm{J} \\ 1\,\mathrm{J} & 0.9\,\mathrm{J} & 0.08\,\mathrm{W} \end{bmatrix}.$$

Although this disk drive can operate with arbitrary thresholds, we consider only two: 0.2 sec and 1.0 sec.

As specified in Section 3.3, the processor and disk drive are each equipped with an RC. Here, we consider two LS, $N = 3$ and $N = 9$. As for the penalty function, after extensive experimentation, we select

$$P\left(\frac{e(k-1)}{w_p(k-1)}\right) = \frac{1 + \mathrm{erf}\left(1100\frac{e(k-1)}{w_p(k-1)} - 0.0029\right)}{2},$$

where

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt.$$

3.5 Results

For the purpose of comparison, we first evaluate the performance of constant parameter strategies, defined by keeping constant, throughout all the jobs, the frequency and voltage of the processor and the threshold of the disk drive. Table 1 summarizes the simulation results, showing the energy consumption, in Joules, for each of the four loads and each of the four constant parameter strategies. For each load, the smallest energy consumption is shown in bold. For instance, for high load, the smallest energy consumption is 23087.6 J, achieved by using the combination of parameters (333 MHz, 1.0 V) and 0.2 sec. The results in Table 1 are used below as a benchmark against which to compare the efficacy of the control architecture of Fig. 1.

Table 1. Energy Consumption for Constant Parameter Strategies

Management strategies	High load $w_p(k)=10^9$	Medium load $w_p(k)=5\text{x}10^8$	Low load $w_p(k)=10^8$	Variable load
(333MHz,1.0V) & 0.2sec	**23087.6**	19273.8	16222.8	19528.4
(333MHz,1.0V) & 1.0sec	28527.6	24713.8	4124.8	19123.0
(733MHz,1.5V) & 0.2sec	27867.9	21664.0	**3488.5**	17674.5
(733MHz,1.5V) & 1.0sec	33307.9	**12162.4**	**3488.5**	**16321.3**

Table 2 summarizes the simulation results in terms of probability of residence in the various states of the RCs. The results are displayed for high, medium, and low loads, and for $N = 3$ and $N = 9$. Each entry of the table is a 2×2 matrix, with rows corresponding to the states of the processor RC ((333 MHz, 1.0 V) and (733 MHz, 1.5 V), respectively), and columns corresponding to the states of the disk drive RC (0.2 sec and 1.0 sec, respectively). The entries of each of these 2×2 matrices are probabilities of residence in the corresponding states. For instance, the $(1, 2)$ entry of each of the matrices is the probability of choosing the combination of parameters: (333 MHz, 1.0 V) for the processor, and 1.0 sec for the disk drive.

Based on Tables 1 and 2, the following observations can be made:

Table 2. Probability of Residence in Various States

Levels of selectivity	High load $w_p(k)$=10^9	Medium load $w_p(k)$=5x10^8	Low load $w_p(k)$=10^8
N=3	$\begin{bmatrix} 0.85 & 0.04 \\ 0.1 & 0.01 \end{bmatrix}$	$\begin{bmatrix} 0.02 & 0.01 \\ 0.02 & 0.95 \end{bmatrix}$	$\begin{bmatrix} 0.18 & 0.19 \\ 0.32 & 0.31 \end{bmatrix}$
N=9	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.99 \end{bmatrix}$	$\begin{bmatrix} 0.07 & 0.09 \\ 0.42 & 0.42 \end{bmatrix}$

Table 3. Energy Consumption for Various Loads

Levels of selectivity	High load $w_p(k)$=10^9	Medium load $w_p(k)$=5x10^8	Low load $w_p(k)$=10^8	Variable load
N=3	23903.1	12598.2	5820.3	14173.9
N=9	23087.6	12165.1	4420.3	13316.1

(a) *A system with RCs exhibits rational behavior.* For a system to be rational, its RCs should most frequently select the combination of parameters that corresponds to the bold entries in Table 1. We see that this is indeed the case. For instance, for medium load, according to Table 1, the best combination of parameters is (733 MHz, 1.5 V) and 1.0 sec. For $N = 3$, Table 2 shows that this combination is selected with probability 0.95.

(b) *Increasing N increases the probability of residing in an optimal state.* This is true even when the optimal state is not unique. For instance, for low load, Table 1 shows that selecting (733 MHz, 1.5 V) for the processor minimizes energy consumption regardless of the threshold of the disk drive. Table 2 shows that, for $N = 3$, the probability of selecting (733 MHz, 1.5 V) for the processor is 0.63, split equally between the two thresholds. However, for $N = 9$, this probability of residence increases to 0.84, and is still split equally.

Table 3 summarizes the simulation results in terms of energy consumption for high, medium, low, and variable loads, and for $N = 3$ and $N = 9$.

Comparing Tables 1 and 3, the following observations can be made:

(c) *Increasing N improves power efficiency.* Obviously, this is a consequence of observation (b) above, since residing more often in an optimal state decreases energy consumption.

(d) *For variable load, RCs outperform all constant parameter strategies.* This is because RCs are capable of adapting to a better frequency and threshold when the load varies. The energy saving achieved by the RCs compared to constant parameter strategies is at least 18.4% and up to 31.8%.

4. RCs IN WIRELESS SENSOR NETWORKS

In this section, we consider an ad hoc wireless sensor network with M sensors spread uniformly in an area A, and a base station receiving information transmitted from the sensors. Each sensor has two modes: active and asleep. A sensor in active mode transmits information directly to the base station, while, in asleep mode, it does not transmit any information to conserve energy. In order to collect information about area A, the base station requires spatially uniform information sent from the sensors. The larger the number of spatially uniformly distributed active sensors, the better the quality of the information. However, if too many sensors are active, the information provided by the sensors may be redundant; their batteries will run out quickly, requiring frequent re-seeding. Moreover, we require the duty cycles of sensors to be roughly the same to maximize the shortest battery life in the network. Hence, it is assumed that the base station requires a certain number, M^*, of spatially uniformly distributed active sensors with duty cycles being roughly the same. The number M^* and the spatial and duty cycle uniformity requirements are referred to as the required *quality of service* (QoS). Below, we solve this problem using ring RCs.

4.1 Modeling

Assume the following:

(i) The system operates in slotted time indexed by $k = 1, 2, 3, \ldots$, with slot duration Δt.

(ii) The system consists of M sensors and the base station requires M^* of them to be active during each time slot.

(iii) Each sensor is equipped with a ring RC (described in more detail below) intended to select its operating mode. The sensors are allowed to change their operating modes at the beginning of a time slot.

(iv) Let $m(k)$ be the number of active sensors during time slot k. During this slot, the base station receives $m(k)$ packets of information, and broadcasts penalties $P_1(m(k))$ and $P_2(m(k))$ for active and asleep sensors, respectively. After extensive experimentation, we have selected P_1 and P_2 as follows:

$$P_1(m(k)) = \begin{cases} 2 & \text{if } 0 \le m(k) \le M^*, \\ 4(m(k) - M^*) + 2 & \text{if } M^* \le m(k) \le M^* + 1, \\ 6 & \text{if } M^* + 1 \le m(k) \le M, \end{cases} \tag{6}$$

and

$$P_2(m(k)) = \begin{cases} 6 & \text{if } 0 \le m(k) \le M^* - 1, \\ -4(m(k) - M^*) + 2 & \text{if } M^* - 1 \le m(k) \le M^* \\ 2 & \text{if } M^* \le m(k) \le M, \end{cases} \tag{7}$$

As we can see from assumption (iv), when the number of active sensors is larger than M^*, active sensors receive more penalty than asleep sensors so that the active sensors are more inclined to switch their operating modes to asleep. Similarly, asleep sensors are more inclined to switch their operating modes to active when the number of active sensors is less than M^*.

4.2 Rational Controllers: Ring RCs

Assume that the sensors are labeled by $i = 1, 2, \ldots, M$ and that the RC corresponding to the ith sensor is referred to as the ith RC and has state $x_i \in X$, where $X = [0, 1)$. Each RC is selected to be a ring RC and assigned an initial condition, uniformly distributed on $(0, 1)$. The ith sensor is active or asleep if $0.5 < x_i < 1$ or $0 \le x_i \le 0.5$, respectively.

Since the system operates in discrete time slots, a discrete time implementation of each ring RC is proposed as:

$$x_i(k+1)=\begin{cases}\dfrac{\zeta}{2} & \text{if } 0\le\{y_i(k+1)\}\le\dfrac{1}{2} \text{ and } \dfrac{1}{2}<x_i(k)<1,\\ \dfrac{1+\zeta}{2} & \text{if } \dfrac{1}{2}<\{y_i(k+1)\}<1 \text{ and } 0\le x_i(k)\le\dfrac{1}{2},\\ \{y_i(k+1)\} & \text{otherwise,}\end{cases} \tag{8}$$

$$y_i(k+1) = x_i(k) + (\phi(x_i(k)))^{N_i}\Delta t', \tag{9}$$

$$\phi(x_i(k)) = \begin{cases} P_1(m(k)) & \text{if } 0.5 < x_i(k) < 1,\\ P_2(m(k)) & \text{if } 0 \le x_i(k) \le 0.5,\end{cases} \tag{10}$$

where N_i is the LS of the ith RC, $\Delta t' \ll \Delta t$ is a small number, and ζ is a random variable uniformly distributed on $(0, 1)$. The random variable ζ is needed so that whenever a sensor switches from active to asleep (or from asleep to active), the state of the corresponding RC resets to a new point uniformly distributed on $(0, 0.5)$ (or on $(0.5, 1)$). This is needed to avoid a state clustering problem discussed in Kabamba et al. (2007a).

4.3 Scenarios

We assume $M = 100$ and $N_i = N$, $i = 1, 2, \ldots, M$, and consider the following two scenarios.

(α) The system is operating from time $k = 1$ to 50000, and the number of spatially uniformly distributed active sensors required is $M^* = 25$.

(β) The systems is operating from time $k = 1$ to 60000. Let the time intervals K_1, K_2, and K_3, be defined by $1 \le k \le 20000$, $20001 \le k \le 40000$ and $40001 \le k \le 60000$, respectively. The number of active sensors required for time intervals K_1, K_2, and K_3 are $M_1^* = 25$, $M_2^* = 75$, and $M_3^* = 35$, respectively.

4.4 Results

Simulations of the system are carried out for scenarios (α) and (β) with $N = 1, 2, 3$ and $N = 3$, respectively, and $\Delta t' = 0.00003$. In order to describe the results of scenario (α), for a given N, introduce

$$F_N = \frac{\sum_{k=1}^{50000} I_N(k)}{50000}, \tag{11}$$

where

$$I_N(k) = \begin{cases}1 & \text{if } M^* - 1 \le m(k) \le M^* + 1\\ 0 & \text{otherwise.}\end{cases} \tag{12}$$

Clearly, the number F_N denotes the frequency with which the system is close to being optimal for a given N. Similarly, to describe the results of scenario (β), for given N and interval K_i ($i = 1, 2, 3$), introduce

$$F_{N,i} = \frac{\sum_{k\in K_i} I_{N,i}(k)}{20000}, \tag{13}$$

where

$$I_{N,i}(k) = \begin{cases}1 & \text{if } M_i^* - 1 \le m(k) \le M_i^* + 1\\ 0 & \text{otherwise.}\end{cases} \tag{14}$$

The results of the simulations for scenario (α) are given in Fig. 2 and Table 4. The numbers of active sensors, $m(k)$, as a function of time are plotted in Fig. 2. The values of F_N and the time required to converge to within 5% of F_N, denoted by T_N, are recorded in Table 4.

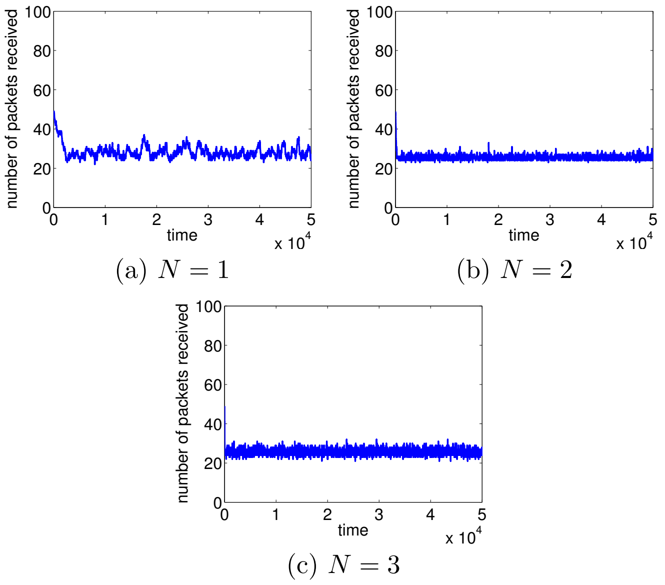


Fig. 2. Number of active sensors $m(k)$ as a function of time for $N = 1$, 2, and 3

Table 4. Function F_N and Convergence Time T_N

	F_N	T_N
N=1	29.91%	34803
N=2	88.94%	8114
N=3	93.37%	407

The following observations are made:

(i) From Fig. 2, the number of active sensors fluctuates around $M^* = 25$ at steady state for all N. Moreover, as indicated in Table 4, the frequency with which the system has $M^* - 1$, M^*, or $M^* + 1$ active sensors increases with N. When $N = 3$, that frequency is 93.37%.

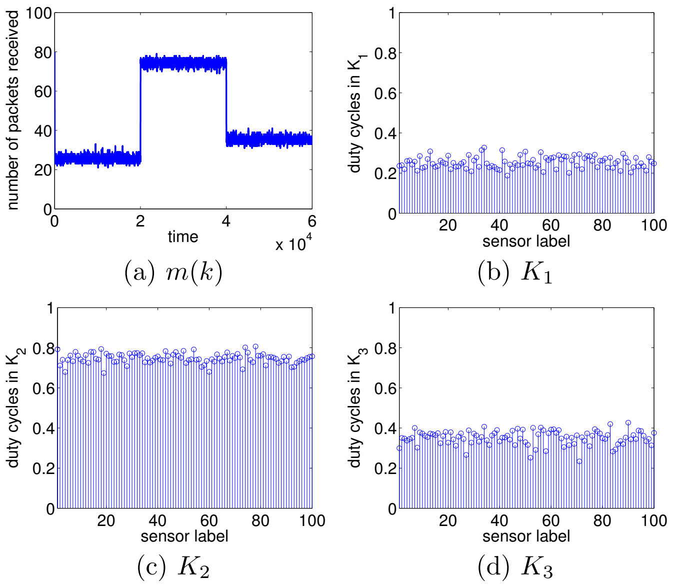
(ii) As shown in Table 4, the time for the system to converge decreases as N becomes large. The convergence time is 407 time steps when $N = 3$.

The results of the simulation for scenario (β) are given in Fig. 3 and Table 5. Fig. 3(a) plots the number of active sensors as a function of time for $N = 3$, while Fig. 3(b)–(d) show the duty cycle each sensor during intervals K_1, K_2, K_3. The values of $F_{N,i}$ and the time required to converge to within 5% of $F_{N,i}$, denoted by $T_{N,i}$, are recorded in Table 5.

Clearly, the system is able to quickly adapt to changing QoS requirement and maintain roughly the same duty cycles for all sensors.

4.5 Discussion

Adaptive QoS has been considered previously in Iyer and Kleinrock (2003), where rational automata with linear



(a) $m(k)$ (b) K_1 (c) K_2 (d) K_3

Fig. 3. Results for scenario (β)

Table 5. Function $F_{N,i}$ and Convergence Time $T_{N,i}$

	K_1		K_2		K_3	
	$F_{N,1}$	$T_{N,1}$	$F_{N,2}$	$T_{N,2}$	$F_{N,3}$	$T_{N,3}$
N=3	93.30%	563	93.67%	463	94.84%	686

tactics (see Tsetlin (1973)) have been used. Convergence to the desired number of active sensors was demonstrated, however, the active sensors remained the same, preventing duty cycle and spatial uniformity. This shortcoming was partially alleviated in Kay and Frolik (2004), where active sensors were not necessarily the same ones. However, neither spatial nor duty cycle uniformity have been demonstrated. In contrast, the current work results in the following features:

(i) the required QoS is achieved with high probabilities for relatively small LS;
(ii) the convergence rate can be made fast by using high LS;
(iii) the duty cycles of the sensors are roughly the same;
(iv) the sensors provide spatially uniform information about the area to the base station (see Kabamba et al. (2007a) for details);
(v) the system is able to rapidly adapt to changes in the required number of active sensors.

5. CONLUSIONS

As it is shown in this paper, rational controllers can be used successfully to determine most efficient regimes for systems operation. Future work in this area will include the development of a comprehensive theory for analysis and design of systems with rational controllers and its applications in various areas of science and engineering, where a resilient performance in unpredictable varying environments is required.

REFERENCES

J. Flinn, P. T. Kabamba, S. M. Meerkov, and C. Y. Tang. Power-improvement capacity of computing and communication devices. Control Group Report CGR #04-01, Department of EECS, University of Michigan, Ann Arbor, MI, 2004.

J. Flinn, P. T. Kabamba, W.-C. Lin, S. M. Meerkov, and C. Y. Tang. Collaborative power management in WPANs using rational controllers: A case study. *Mathematical Problems in Engineering*, 2005(5):491–501, 2005.

Intel Corporation. Intel 80200 processor based on Intel XScale microarchitecture. `http://www.intel.com/design/iio/manuals/273411.htm`, 2005.

R. Iyer and L. Kleinrock. QoS control for sensor networks. In *Proceedings of the IEEE International Conference on Communications*, volume 1, pages 517–521, 2003.

P. T. Kabamba, W.-C. Lin, and S. M. Meerkov. Rational controllers in sensor networks: QoS and coordination. Control Group Report CGR #07-05, Department of EECS, University of Michigan, Ann Arbor, MI, 2007a.

P. T. Kabamba, W.-C. Lin, and S. M. Meerkov. Rational probabilistic deciders-Part I: Individual behavior. *Mathematical Problems in Engineering*, 2007:Article ID 35897, 31 pages, 2007b.

P. T. Kabamba, W.-C. Lin, and S. M. Meerkov. Rational probabilistic deciders-Part II: Collective behavior. *Mathematical Problems in Engineering*, 2007:Article ID 82184, 34 pages, 2007c.

J. Kay and J. Frolik. Quality of Service analysis and control for wireless sensor networks. In *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 359–368, 2004.

S. M. Meerkov. Mathematical theory of behavior-individual and collective behavior of retardable elements. *Mathematical Biosciences*, 43(1-2):41–106, 1979.

K. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, 1989.

C. Pereira, V. Raghunathan, S. Gupta, R. Gupta, and M. Srivastava. A software architecture for building power aware real time operating systems. Technical Report #02-07, Department of Information and Computer Science, University of California, Irvine, CA, 2002.

M. L. Tsetlin. *Automata Theory and Modeling of Biological Systems.* New York: Academic, 1973.

V. I. Varshavsky. *Collective Behavior of Automata.* Nauka, Moscow, 1973. (in Russian).

Appendix A. LIST OF ABBREVIATIONS

LS	Level of Selectivity
PD	Probabilistic Decider
QoS	Quality of Service
RC	Rational Controller
RPD	Rational Probabilistic Decider
SN	Sensor Network
WPAN	Wireless Personal Area Network