Chapter 9

Low Power Dynamic Scheduling for Computing Systems

Michael J. Neely
University of Southern California, mjneely@usc.edu

This is a book chapter from a 2012 CRC Press book. The page numbering in this document is different from the page numbering in the book. The proper citation is:
M. J. Neely. (2012) Low power dynamic scheduling for computing systems. In F. R. Yu, X. Zhang, & V. C. M. Leung (Eds.), *Green Communications and Networking* (pp. 219-259), CRC Press.

This chapter considers energy-aware control for a computing system with two states: *active* and *idle*. In the active state, the controller chooses to perform a single task using one of multiple task processing modes. The controller then saves energy by choosing an amount of time for the system to be idle. These decisions affect processing time, energy expenditure, and an abstract *attribute vector* that can be used to model other criteria of interest (such as processing quality or distortion). The goal is to optimize time average system performance. Applications of this model include a smart phone that makes energy-efficient computation

and transmission decisions, a computer that processes tasks subject to rate, quality, and power constraints, and a smart grid energy manager that allocates resources in reaction to a time varying energy price.

The solution methodology of this chapter uses the theory of *optimization for renewal systems* developed in [25][30]. Section 9.1 focuses on a computer system that seeks to minimize average power subject to processing rate constraints for different classes of tasks. Section 9.2 generalizes to treat optimization for a larger class of systems. Section 9.3 extends the model to allow control actions to react to a random event observed at the beginning of each active period, such as a vector of current channel conditions or energy prices.

9.1 Task Scheduling with Processing Rate Constraints

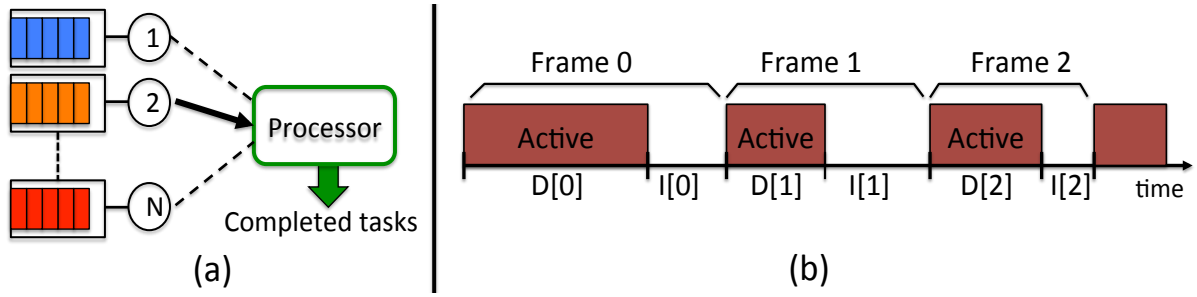

Figure 9.1: (a) A processor that chooses from one of N classes on each frame k. (b) A timeline illustrating the active and idle periods for each frame.

To illustrate the method, this section considers a particular system. Consider a computer system that repeatedly processes tasks. There are N classes of tasks, where N is a positive integer. For simplicity, assume each class always has a new task ready to be performed (this is extended to randomly arriving tasks in Section 9.2.3). The system operates over time intervals called *frames*. Each frame $k \in \{0, 1, 2, \ldots\}$ begins with an *active period* of size $D[k]$ and ends with an *idle period* of size $I[k]$ (see Fig. 9.1). At the beginning of each active period k, the controller selects a new task of class $c[k] \in \{1, \ldots, N\}$. It also chooses a *processing mode* $m[k]$ from a finite set $\mathcal{M}$ of possible processing options. These control decisions affect the duration $D[k]$ of the active period and the energy $e[k]$ that is incurred. The controller then selects an amount of time $I[k]$ to be idle, where $I[k]$ is chosen such that

$0 \leq I[k] \leq I_{max}$ for some positive number I_{max}. Choosing $I[k] = 0$ effectively skips the idle period, so there can be back-to-back active periods. For simplicity, this section assumes that no energy is expended in the idle state.

Assume that $D[k]$ and $e[k]$ are *random functions* of the class and mode decisions for frame k. Specifically, assume $D[k]$ and $e[k]$ are conditionally independent of the past, given the current $(c[k], m[k])$ that is used, with mean values given by functions $\hat{D}(c, m)$ and $\hat{e}(c, m)$ defined over $(c, m) \in \{1, \ldots, N\} \times \mathcal{M}$:

$$\hat{D}(c,m) \triangleq \mathbb{E}\left[D[k]|(c[k], m[k]) = (c, m)\right] \quad , \quad \hat{e}(c,m) \triangleq \mathbb{E}\left[e[k]|(c[k], m[k]) = (c, m)\right]$$

where the notation "$a \triangleq b$" represents "a is defined to be equal to b." This chapter uses $\hat{D}(c[k], m[k])$ and $\hat{e}(c[k], m[k])$ to denote expectations given a particular decision $(c[k], m[k])$ for frame k:

$$\hat{D}(c[k], m[k]) = \mathbb{E}\left[D[k]|c[k], m[k]\right] \quad , \quad \hat{e}(c[k], m[k]) = \mathbb{E}\left[e[k]|c[k], m[k]\right]$$

It is assumed there is a positive value D_{min} such that $D[k] \geq D_{min}$ for all frames k, regardless of the $(c[k], m[k])$ decisions. Thus, all frame sizes are at least D_{min} units of time. Further, for technical reasons, it is assumed that second moments of $D[k]$ and $e[k]$ are bounded by a finite constant σ^2, so that:

$$\mathbb{E}\left[D[k]^2\right] \leq \sigma^2 \quad , \quad \mathbb{E}\left[e[k]^2\right] \leq \sigma^2 \tag{9.1}$$

where (9.1) holds regardless of the policy for selecting $(c[k], m[k])$. The conditional joint distribution of $(D[k], e[k])$, given $(c[k], m[k])$, is otherwise arbitrary, and only the mean values $\hat{D}(c, m)$ and $\hat{e}(c, m)$ are known for each $c \in \{1, \ldots, N\}$ and $m \in \mathcal{M}$. In the special case of a deterministic system, the functions $\hat{D}(c, m)$ and $\hat{e}(c, m)$ can be viewed as deterministic mappings from a given control action $(c[k], m[k]) = (c, m)$ to the actual delay $D[k] = \hat{D}(c, m)$ and energy $e[k] = \hat{e}(c, m)$ experienced on frame k, rather than as expectations of these values.

9.1.1 Examples of Energy-Aware Processing

Consider an example where a computer system performs computation for different tasks. Suppose the system uses a chip multi-processor that can select between one of multiple processing modes for each task. For example, this might be done using voltage/frequency scaling, or by using a choice of different processing cores [9][2]. Let $\mathcal{M}$ represent the set of processing modes. For each mode $m \in \mathcal{M}$, define:

$$\begin{aligned} T_{setup}(m) &\triangleq \text{Setup time for mode } m. \\ e_{setup}(m) &\triangleq \text{Setup energy for mode } m. \\ DPI(m) &\triangleq \text{Average delay-per-instruction for mode } m. \\ EPI(m) &\triangleq \text{Average energy-per-instruction for mode } m. \end{aligned}$$

Further suppose that tasks of class $c \in \{1, \ldots, N\}$ have an average number of instructions equal to $\overline{S}_c$. For simplicity, suppose the number of instructions in a task is independent of the energy and delay of each individual instruction. Then the average energy and delay functions $\hat{e}(c, m)$ and $\hat{D}(c, m)$ are:

$$\begin{aligned} \hat{e}(c, m) &= e_{setup}(m) + \overline{S}_c EPI(m) \\ \hat{D}(c, m) &= T_{setup}(m) + \overline{S}_c DPI(m) \end{aligned}$$

In cases when the system cannot be modeled using $DPI(m)$ and $EPI(m)$ functions, the expectations $\hat{e}(c, m)$ and $\hat{D}(c, m)$ can be estimated as empirical averages of energy and delay observed when processing type c tasks with mode m. While this example assumes all classes have the same set of processing mode options $\mathcal{M}$, this can easily be extended to restrict each class c to its own subset of options $\mathcal{M}_c$.

As another example, consider the problem of *wireless data transmission*. Here, each task represents a packet of data that must be transmitted. Let $\mathcal{M}$ represent the set of wireless transmission options (such as modulation and coding strategies). For each $m \in \mathcal{M}$, define $\mu(m)$ as the transmission rate (in bits per unit time) under option m, and let $power(m)$ be the power used. For simplicity, assume there are no transmission errors. Let $\overline{B}_c$ represent

the average packet size for class $c \in \{1, \ldots, N\}$, in units of bits. Thus:

$$\begin{aligned} \hat{e}(c,m) &= power(m)\overline{B}_c/\mu(m) \\ \hat{D}(c,m) &= \overline{B}_c/\mu(m) \end{aligned}$$

In the case when each transmission mode $m \in \mathcal{M}$ has a known error probability, the functions $\hat{e}(c,m)$ and $\hat{D}(c,m)$ can be redefined to account for retransmissions. An important alternative scenario is when channel states are time-varying but can be measured at the beginning of each frame. This can be treated using the extended theory in Section 9.3.

9.1.2 Time Averages as Ratios of Frame Averages

The goal is to design a control policy that makes decisions over frames to minimize time average power subject to processing each class $n \in \{1, \ldots, N\}$ with rate at least λ_n, for some desired processing rates $(\lambda_1, \ldots, \lambda_N)$ that are given. Before formalizing this as a mathematical optimization, this subsection shows how to write time averages in terms of frame averages. Suppose there is a control policy that yields a sequence of energies $\{e[0], e[1], e[2], \ldots\}$ and corresponding frame sizes $\{D[0]+I[0], D[1]+I[1], D[2]+I[2], \ldots\}$ for each frame $k \in \{0, 1, 2, \ldots\}$. The *frame averages* $\overline{e}$, $\overline{D}$, $\overline{I}$ are defined:

$$\overline{e} \triangleq \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} e[k] \quad , \quad \overline{D} \triangleq \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} D[k] \quad , \quad \overline{I} \triangleq \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} I[k] \tag{9.2}$$

where, for simplicity, it is assumed the limits converge to constants with probability 1. Note that $\overline{e}$ does *not* represent the time average power used by the system, because it does not consider the amount of time spent in each frame. The time average power considers the accumulated energy used divided by the total time, and is written as follows:

$$\lim_{K\to\infty} \frac{\sum_{k=0}^{K-1} e[k]}{\sum_{k=0}^{K-1}(D[k]+I[k])} = \lim_{K\to\infty} \frac{\frac{1}{K}\sum_{k=0}^{K-1} e[k]}{\frac{1}{K}\sum_{k=0}^{K-1}(D[k]+I[k])} = \frac{\overline{e}}{\overline{D}+\overline{I}}$$

Therefore, the time average power is equal to the average energy per frame divided by the average frame size. This simple observation is often used in *renewal-reward theory* [6][35].

For each class $n \in \{1, \ldots, N\}$ and each frame k, define an *indicator variable* $1_n[k]$ that is 1 if the controller chooses to process a class n task on frame k, and 0 else:

$$1_n[k] \triangleq \begin{cases} 1 & \text{if } c[k] = n \\ 0 & \text{if } c[k] \neq n \end{cases}$$

Then $\overline{1}_n \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} 1_n[k]$ is the fraction of frames that choose class n, and the ratio $\overline{1}_n/(\overline{D} + \overline{I})$ is the time average rate of processing class n tasks, in tasks per unit time.

The problem of minimizing time average power subject to processing each class n at a rate of at least λ_n tasks per unit time is then mathematically written as follows:

$$\text{Minimize:} \quad \frac{\overline{e}}{\overline{D} + \overline{I}} \tag{9.3}$$

$$\text{Subject to:} \quad \frac{\overline{1}_n}{\overline{D} + \overline{I}} \geq \lambda_n \quad \forall n \in \{1, \ldots, N\} \tag{9.4}$$

$$(c[k], m[k]) \in \{1, \ldots, N\} \times \mathcal{M} \quad \forall k \in \{0, 1, 2, \ldots\} \tag{9.5}$$

$$0 \leq I[k] \leq I_{max} \quad \forall k \in \{0, 1, 2, \ldots\} \tag{9.6}$$

where the objective (9.3) is average power, the constraint (9.4) ensures the processing rate of each class n is at least λ_n, and constraints (9.5)-(9.6) ensure that $c[k] \in \{1, \ldots, N\}$, $m[k] \in \mathcal{M}$, and $0 \leq I[k] \leq I_{max}$ for each frame k.

9.1.3 Relation to Frame Average Expectations

The problem (9.3)-(9.6) is defined by frame averages. This subsection shows that frame averages are related to frame average *expectations*, and hence can be related to the expectation functions $\hat{D}(c, m)$, $\hat{e}(c, m)$. Consider any (possibly randomized) control algorithm for selecting $(c[k], m[k])$ over frames, and assume this gives rise to well defined expectations $\mathbb{E}[e[k]]$ for each frame k. By the law of iterated expectations, it follows that for any given frame $k \in \{0, 1, 2, \ldots\}$:

$$\mathbb{E}[e[k]] = \mathbb{E}[\,\mathbb{E}[e[k]|c[k], m[k]]\,] = \mathbb{E}[\,\hat{e}(c[k], m[k])\,] \tag{9.7}$$

Furthermore, because second moments are bounded by a constant σ^2 on each frame k, the *bounded moment convergence theorem* (given in Appendix A) ensures that if the frame average energy converges to a constant $\overline{e}$ with probability 1, as defined in (9.2), then $\overline{e}$ is the same as the frame average expectation:

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}e[k] = \overline{e} = \lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\hat{e}(c[k],m[k])\right]$$

The same goes for the quantities $\overline{D}$, $\overline{I}$, $\overline{1}_n$. Hence, one can interpret the problem (9.3)-(9.6) using frame average expectations, rather than pure frame averages.

9.1.4 An Example with One Task Class

Consider a simple example with only one type of task. The system processes a new task of this type at the beginning of every busy period. The energy and delay functions can then be written purely in terms of the processing mode $m \in \mathcal{M}$, so that we have $\hat{e}(m)$ and $\hat{D}(m)$. Suppose there are only two processing mode options, so that $\mathcal{M} = \{1, 2\}$, and that each option leads to a deterministic energy and delay, as given below:

$$m[k] = 1 \implies (e[k], D[k]) = (\hat{e}(1), \hat{D}(1)) = (1, 7) \tag{9.8}$$

$$m[k] = 2 \implies (e[k], D[k]) = (\hat{e}(2), \hat{D}(2)) = (3, 4) \tag{9.9}$$

Option $m[k] = 1$ requires 1 unit of energy but 7 units of processing time. Option $m[k] = 2$ is more energy-expensive (requiring 3 units of energy) but is faster (taking only 4 time units). The idle time $I[k]$ is chosen every frame in the interval $[0, 10]$, so that $I_{max} = 10$.

No constraints

For this system, suppose we seek to minimize average power $\overline{e}/(\overline{D} + \overline{I})$, with no processing rate constraint. Consider three possible algorithms:

1. Always use $m[k] = 1$ and $I[k] = I$ for all frames k, for some constant $I \in [0, 10]$.

2. Always use $m[k] = 2$ and $I[k] = I$ for all frames k, for some constant $I \in [0, 10]$.

3. Always use $I[k] = I$ for all frames k, for some constant $I \in [0, 10]$. However, each frame k, independently choose $m[k] = 1$ with probability p, and $m[k] = 2$ with probability $1 - p$ (for some p that satisfies $0 \leq p \leq 1$).

Clearly the third algorithm contains the first two for $p = 1$ and $p = 0$, respectively. The time average power under each algorithm is:

$$\begin{aligned}
&m[k] = 1 \text{ always} \quad \Longrightarrow \quad \frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{\hat{e}(1)}{\hat{D}(1) + I} = \frac{1}{7 + I} \\
&m[k] = 2 \text{ always} \quad \Longrightarrow \quad \frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{\hat{e}(2)}{\hat{D}(2) + I} = \frac{3}{4 + I} \\
&\text{probabilistic rule} \quad \Longrightarrow \quad \frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{p\hat{e}(1) + (1-p)\hat{e}(2)}{p\hat{D}(1) + (1-p)\hat{D}(2) + I} = \frac{1p + 3(1-p)}{7p + 4(1-p) + I}
\end{aligned}$$

It is clear that in all three cases, we should choose $I = 10$ to minimize average power. Further, it is clear that always choosing $m[k] = 1$ is better than always choosing $m[k] = 2$. However, it is not immediately obvious if a randomized mode selection rule can do even better. The answer is no: In this case, power is minimized by choosing $m[k] = 1$ and $I[k] = 10$ for all frames k, yielding average power $1/17$.

This fact holds more generally: Let $a(\alpha)$ and $b(\alpha)$ be any deterministic real valued functions defined over general *actions* α that are chosen in an abstract *action space* $\mathcal{A}$. Assume the functions are bounded, and that there is a value $b_{min} > 0$ such that $b(\alpha) \geq b_{min}$ for all $\alpha \in \mathcal{A}$. Consider designing a randomized choice of α that minimizes $\mathbb{E}[a(\alpha)]/\mathbb{E}[b(\alpha)]$, where the expectations are with respect to the randomness in the α selection. The next lemma shows this is done by *deterministically* choosing an action $\alpha \in \mathcal{A}$ that minimizes $a(\alpha)/b(\alpha)$.

Lemma 1. *Under the assumptions of the preceding paragraph, for any randomized choice of $\alpha \in \mathcal{A}$ we have:*

$$\frac{\mathbb{E}[a(\alpha)]}{\mathbb{E}[b(\alpha)]} \geq \inf_{\alpha \in \mathcal{A}} \left[\frac{a(\alpha)}{b(\alpha)}\right]$$

and so the infimum of the ratio over the class of deterministic decisions yields a value that is less than or equal to that of any randomized selection.

Proof. Consider any randomized policy that yields expectations $\mathbb{E}[a(\alpha)]$ and $\mathbb{E}[b(\alpha)]$. Without loss of generality, assume these expectations are achieved by a policy that randomizes over a finite set of M actions $\alpha_1, \ldots, \alpha_M$ in $\mathcal{A}$ with some probabilities $p_1, \ldots, p_M$:[1]

$$\mathbb{E}[a(\alpha)] = \sum_{m=1}^{M} p_m a(\alpha_m) \ , \ \mathbb{E}[b(\alpha)] = \sum_{m=1}^{M} p_m b(\alpha_m)$$

Then, because $b(\alpha_m) > 0$ for all α_m, we have:

$$\begin{aligned}
\frac{\mathbb{E}[a(\alpha)]}{\mathbb{E}[b(\alpha)]} &= \frac{\sum_{m=1}^{M} p_m a(\alpha_m)}{\sum_{m=1}^{M} p_m b(\alpha_m)} \\
&= \frac{\sum_{m=1}^{M} p_m b(\alpha_m)[a(\alpha_m)/b(\alpha_m)]}{\sum_{m=1}^{M} p_m b(\alpha_m)} \\
&\geq \frac{\sum_{m=1}^{M} p_m b(\alpha_m) \inf_{\alpha \in \mathcal{A}}[a(\alpha)/b(\alpha)]}{\sum_{m=1}^{M} p_m b(\alpha_m)} \\
&= \inf_{\alpha \in \mathcal{A}}[a(\alpha)/b(\alpha)]
\end{aligned}$$

□

One Constraint

The preceding subsection shows that unconstrained problems can be solved by deterministic actions. This is not true for constrained problems. This subsection shows that adding just a single constraint often *necessitates* the use of randomized actions. Consider the same problem as before, with two choices for $m[k]$, and with the same $\hat{e}(m)$ and $\hat{D}(m)$ values given in (9.8)-(9.9). We want to minimize $\overline{e}/(\overline{D} + \overline{I})$ subject to $1/(\overline{D} + \overline{I}) \geq 1/5$, where $1/(\overline{D} + \overline{I})$ is the rate of processing jobs. The constraint is equivalent to $\overline{D} + \overline{I} \leq 5$.

Assume the algorithm chooses $I[k]$ over frames to yield an average $\overline{I}$ that is somewhere in

[1] Indeed, because the set $\mathcal{S} = \{(a(\alpha), b(\alpha)) \text{ such that } \alpha \in \mathcal{A}\}$ is bounded, the expectation $(\mathbb{E}[a(\alpha)], \mathbb{E}[b(\alpha)])$ is finite and is contained in the convex hull of $\mathcal{S}$. Thus, $(\mathbb{E}[a(\alpha)], \mathbb{E}[b(\alpha)])$ is a convex combination of a finite number of points in $\mathcal{S}$.

the interval $0 \leq \overline{I} \leq 10$. Now consider different algorithms for selecting $m[k]$. If we choose $m[k] = 1$ always (so that $(\hat{e}(1), \hat{D}(1)) = (1, 7)$), then:

$$m[k] = 1 \text{ always} \quad \Longrightarrow \quad \frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{1}{7 + \overline{I}} \quad , \quad \overline{D} + \overline{I} = 7 + \overline{I}$$

and so it is impossible to meet the constraint $\overline{D} + \overline{I} \leq 5$, because $7 + \overline{I} > 5$.

If we choose $m[k] = 2$ always (so that $(\hat{e}(2), \hat{D}(2)) = (3, 4)$), then:

$$m[k] = 2 \text{ always} \quad \Longrightarrow \quad \frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{3}{4 + \overline{I}} \quad , \quad \overline{D} + \overline{I} = 4 + \overline{I}$$

It is clear that we can meet the constraint by choosing $\overline{I}$ so that $0 \leq \overline{I} \leq 1$, and power is minimized in this setting by using $\overline{I} = 1$. This can be achieved, for example, by using $I[k] = 1$ for all frames k. This meets the processing rate constraint with equality: $\overline{D} + \overline{I} = 4 + 1 = 5$. Further, it yields average power $\overline{e}/(\overline{D} + \overline{I}) = 3/5 = 0.6$.

However, it is possible to reduce average power while also meeting the constraint with equality by using the following randomized policy (which can be shown to be optimal): Choose $I[k] = 0$ for all frames k, so that $\overline{I} = 0$. Then every frame k, independently choose $m[k] = 1$ with probability $1/3$, and $m[k] = 2$ with probability $2/3$. We then have:

$$\overline{D} + \overline{I} = (1/3)7 + (2/3)4 + 0 = 5$$

and so the processing rate constraint is met with equality. However, average power is:

$$\frac{\overline{e}}{\overline{D} + \overline{I}} = \frac{(1/3)1 + (2/3)3}{(1/3)7 + (2/3)4 + 0} = 7/15 \approx 0.466667$$

This is a significant savings over the average power of 0.6 from the deterministic policy.

9.1.5 The Linear Fractional Program for Task Scheduling

Now consider the general problem (9.3)-(9.6) for minimizing average power subject to average processing rate constraints for each of the N classes. Assume the problem is *feasible*, so that it

is possible to choose actions $(c[k], m[k], I[k])$ over frames to meet the desired constraints (9.4)-(9.6). It can be shown that an optimal solution can be achieved over the class of *stationary and randomized policies* with the following structure: Every frame, independently choose vector $(c[k], m[k])$ with some probabilities $p(c,m) = Pr[(c[k], m[k]) = (c,m)]$. Further, use a constant idle time $I[k] = I$ for all frames k, for some constant I that satisfies $0 \leq I \leq I_{max}$. Thus, the problem can be written as the following *linear fractional program* with unknowns $p(c,m)$ and I and known constants $\hat{e}(c,m)$, $\hat{D}(c,m)$, λ_n, and I_{max}:

$$\text{Minimize:} \quad \frac{\sum_{c=1}^{N}\sum_{m\in\mathcal{M}} p(c,m)\hat{e}(c,m)}{I + \sum_{c=1}^{N}\sum_{m\in\mathcal{M}} p(c,m)\hat{D}(c,m)} \tag{9.10}$$

$$\text{Subject to:} \quad \frac{\sum_{m\in\mathcal{M}} p(n,m)}{I + \sum_{c=1}^{N}\sum_{m\in\mathcal{M}} p(c,m)\hat{D}(c,m)} \geq \lambda_n \quad \forall n \in \{1,\ldots,N\} \tag{9.11}$$

$$0 \leq I \leq I_{max} \tag{9.12}$$

$$p(c,m) \geq 0 \quad \forall c \in \{1,\ldots,N\}, \ \forall m \in \mathcal{M} \tag{9.13}$$

$$\sum_{c=1}^{N}\sum_{m\in\mathcal{M}} p(c,m) = 1 \tag{9.14}$$

where the numerator and denominator in (9.10) are equal to $\overline{e}$ and $\overline{D}+\overline{I}$, respectively, under this randomized algorithm, the numerator in the left-hand-side of (9.11) is equal to $\overline{1}_n$, and the constraints (9.13)-(9.14) specify that $p(c,m)$ must be a valid probability mass function.

Linear fractional programs can be solved in several ways. One method uses a nonlinear change of variables to map the problem to a convex program [3]. However, this method does not admit an *online* implementation, because time averages are not preserved through the nonlinear change of variables. Below, an online algorithm is presented that makes decisions every frame k. The algorithm is *not* a stationary and randomized algorithm as described above. However, it yields time averages that satisfy the desired constraints of the problem (9.3)-(9.6), with a time average power expenditure that can be pushed arbitrarily close to the optimal value. A significant advantage of this approach is that it extends to treat cases with random task arrivals, without requiring knowledge of the $(\lambda_1,\ldots,\lambda_N)$ arrival rates, and to treat other problems with observed random events, without requiring knowledge of the probability distribution for these events. These extensions are shown in later sections.

For later analysis, it is useful to write (9.10)-(9.14) in a simpler form. Let $power^{opt}$ be the optimal time average power for the above linear fractional program, achieved by some probability distribution $p^*(c,m)$ and idle time I^* that satisfies $0 \leq I^* \leq I_{max}$. Let $(c^*[k], m^*[k], I^*[k])$ represent the frame k decisions under this stationary and randomized policy. Thus:

$$\frac{\mathbb{E}\left[\hat{e}(c^*[k], m^*[k])\right]}{\mathbb{E}\left[I^*[k] + \hat{D}(c^*[k], m^*[k])\right]} = power^{opt} \tag{9.15}$$

$$\frac{\mathbb{E}\left[1_n^*[k]\right]}{\mathbb{E}\left[I^*[k] + \hat{D}(c^*[k], m^*[k])\right]} \geq \lambda_n \quad \forall n \in \{1, \ldots, N\} \tag{9.16}$$

where $1_n^*[k]$ is an indicator function that is 1 if $c^*[k] = n$, and 0 else. The numerator and denominator of (9.15) correspond to those of (9.10). Likewise, the constraint (9.16) corresponds to (9.11).

9.1.6 Virtual Queues

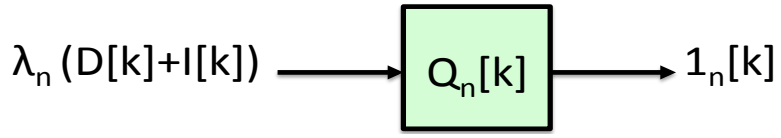

Figure 9.2: An illustration of the virtual queue $Q_n[k]$ from equation (9.18).

To solve the problem (9.3)-(9.6), we first consider the constraints (9.4), which are equivalent to the constraints:

$$\lambda_n(\overline{D} + \overline{I}) \leq \overline{1}_n \quad \forall n \in \{1, \ldots, N\} \tag{9.17}$$

For each constraint $n \in \{1, \ldots, N\}$, define a *virtual queue* $Q_n[k]$ that is updated on frames $k \in \{0, 1, 2, \ldots\}$ by:

$$Q_n[k+1] = \max[Q_n[k] + \lambda_n(D[k] + I[k]) - 1_n[k], 0] \tag{9.18}$$

The initial condition $Q_n[0]$ can be any non-negative value. For simplicity, it is assumed throughout that $Q_n[0] = 0$ for all $n \in \{1, \ldots, N\}$. The update (9.18) can be viewed as a

discrete time queueing equation, where $Q_n[k]$ is the backlog on frame k, $\lambda_n(D[k] + I[k])$ is an effective amount of "new arrivals," and $1_n[k]$ is the amount of "offered service" (see Fig. 9.2). The intuition is that if all virtual queues $Q_n[k]$ are *stable*, then the average "arrival rate" $\lambda_n(\overline{D} + \overline{I})$ must be less than or equal to the average "service rate" $\overline{1}_n$, which ensures the desired constraint (9.17). This is made precise in the following lemma.

Lemma 2. *(Virtual Queues) Suppose $Q_n[k]$ has update equation given by (9.18), with any non-negative initial condition.*

a) For all $K \in \{1, 2, 3, \ldots\}$ we have:

$$\frac{1}{K}\sum_{k=0}^{K-1}[\lambda_n(D[k] + I[k]) - 1_n[k]] \leq \frac{Q_n[K] - Q_n[0]}{K} \tag{9.19}$$

b) If $\lim_{K\to\infty} Q_n[K]/K = 0$ with probability 1, then:

$$\limsup_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1}[\lambda_n(D[k] + I[k]) - 1_n[k]] \leq 0 \quad \textit{with probability 1} \tag{9.20}$$

c) If $\lim_{K\to\infty} \mathbb{E}\left[Q_n[K]\right]/K = 0$, then:

$$\limsup_{K\to\infty}[\lambda_n(\overline{D}[K] + \overline{I}[K]) - \overline{1}_n[K]] \leq 0 \tag{9.21}$$

where $\overline{D}[K]$, $\overline{I}[K]$, $\overline{1}_n[K]$ are defined:

$$\overline{D}[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[D[k]\right] \;\;,\;\; \overline{I}[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[I[k]\right] \;\;,\;\; \overline{1}_n[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[1_n[k]\right]$$

Proof. From (9.18) we have for all $k \in \{0, 1, 2, \ldots\}$:

$$Q_n[k+1] \geq Q_n[k] + \lambda_n(D[k] + I[k]) - 1_n[k]$$

Fixing a positive integer K and summing the above over $k \in \{0, \ldots, K-1\}$ yields:

$$Q_n[K] - Q_n[0] \geq \sum_{k=0}^{K-1} [\lambda_n(D[k] + I[k]) - 1_n[k]]$$

Dividing the above by K proves part (a). Part (b) follows from (9.19) by taking a lim sup. Part (c) follows by first taking expectations of (9.19) and then taking a lim sup. □

Inequality (9.19) shows that the value $Q_n[K]/K$ bounds the amount by which the desired constraint for class n is violated by the time averages achieved over the first K frames. Suppose that $D[k]$, $I[k]$, and $1_n[k]$ have frame averages that converge to constants $\overline{D}$, $\overline{I}$, $\overline{1}_n$ with probability 1. Part (c) of Lemma 2 indicates that if $\lim_{K\to\infty} \mathbb{E}\left[Q_n[K]\right]/K = 0$ for all $n \in \{1, \ldots, N\}$, then $\lambda_n(\overline{D} + \overline{I}) \leq \overline{1}_n$ for all $n \in \{1, \ldots, N\}$.

In the language of queueing theory, a discrete time queue $Q[k]$ is said to be *rate stable* if $\lim_{k\to\infty} Q[k]/k = 0$ with probability 1, and is *mean rate stable* if $\lim_{k\to\infty} \mathbb{E}\left[Q[k]\right]/k = 0$ [25]. With this terminology, the above lemma shows that if $Q_n[k]$ is rate stable then (9.20) holds, and if $Q_n[k]$ is mean rate stable then (9.21) holds.

9.1.7 The Drift-Plus-Penalty Ratio

To stabilize the queues while minimizing time average power, we use *Lyapunov optimization theory*, which gives rise to the *drift-plus-penalty ratio algorithm* [25]. First define $L[k]$ as the sum of the squares of all queues on frame k (divided by 2 for convenience later):

$$L[k] \triangleq \frac{1}{2} \sum_{n=1}^{N} Q_n[k]^2$$

$L[k]$ is often called a *Lyapunov function*, and acts as a scalar measure of the size of the queues. Intuitively, keeping $L[k]$ small leads to stable queues, and we should take actions that tend to shrink $L[k]$ from one frame to the next. Define $\Delta[k]$ as the *Lyapunov drift*,

being the difference in the Lyapunov function from one frame to the next:

$$\Delta[k] \triangleq L[k+1] - L[k]$$

Taking actions to minimize $\Delta[k]$ every frame can be shown to ensure the desired constraints are satisfied whenever it is possible to satisfy them, but does not incorporate power minimization. To incorporate this, every frame k we observe the current queue vector $\boldsymbol{Q}[k] = (Q_1[k], \ldots, Q_N[k])$ and choose control actions $(c[k], m[k], I[k])$ to minimize a bound on the following *drift-plus-penalty ratio*:

$$\frac{\mathbb{E}\left[\Delta[k] + Ve[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]}$$

where V is a non-negative parameter that weights the extent to which power minimization is emphasized. The intuition is that the numerator incorporates both drift and energy. The denominator "normalizes" this by the expected frame size, with the understanding that average power must include both energy and frame size. We soon show that this intuition is correct, in that all desired time average constraints are satisfied, and the time average power is within $O(1/V)$ of the optimal value $power^{opt}$. Hence, average power can be pushed arbitrarily close to optimal by using a sufficiently large value of V. The tradeoff is that average queue sizes grow with V, which impacts the convergence time required to satisfy the desired constraints.

The drift-plus-penalty ratio method was first developed for the context of restless bandit systems in [18][19]. The method was used for optimization of renewal systems in [25][30], which treat problems similar to those considered in this chapter. In the special case when all frame sizes are fixed and equal to one unit of time (a *time slot*), and when $V = 0$, the method reduces to observing queues $\boldsymbol{Q}[k]$ every slot k and taking actions to minimize a bound on $\mathbb{E}\left[\Delta[k]|\boldsymbol{Q}[k]\right]$. This is the rule that generates the classic max-weight scheduling algorithms for queue stability (without performance optimization), developed by Tassiulas and Ephremides in [38][39]. For systems with unit time slots but with $V > 0$, the *drift-plus-penalty ratio* technique reduces to the *drift-plus-penalty* technique of [24][8][29], which treats joint queue stability and penalty minimization in systems with unit size slots.

Bounding the Drift-Plus-Penalty Ratio

To construct an explicit algorithm, we first bound the drift-plus-penalty ratio.

Lemma 3. *For all frames $k \in \{0, 1, 2, \ldots\}$, all possible $\boldsymbol{Q}[k]$, and under any decisions for $(c[k], m[k], I[k])$, we have:*

$$\frac{\mathbb{E}\left[\Delta[k] + Ve[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]} \leq \frac{B}{\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]} + \frac{\mathbb{E}\left[V\hat{e}(c[k], m[k])|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c[k], m[k]) + I[k]|\boldsymbol{Q}[k]\right]} + \frac{\sum_{n=1}^{N} Q_n[k]\mathbb{E}\left[\lambda_n(\hat{D}(c[k], m[k]) + I[k]) - 1_n[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c[k], m[k]) + I[k]|\boldsymbol{Q}[k]\right]} \quad (9.22)$$

where B is a constant that satisfies the following for all possible $\boldsymbol{Q}[k]$ and all policies:

$$B \geq \tfrac{1}{2}\sum_{n=1}^{N}\mathbb{E}\left[(\lambda_n(D[k] + I[k]) - 1_n[k])^2|\boldsymbol{Q}[k]\right]$$

Such a constant B exists by the second moment boundedness assumptions (9.1).

Proof. Note by iterated expectations that (similar to (9.7)):[2]

$$\mathbb{E}\left[D[k]|\boldsymbol{Q}[k]\right] = \mathbb{E}\left[\hat{D}(c[k], m[k])|\boldsymbol{Q}[k]\right] \ , \ \mathbb{E}\left[e[k]|\boldsymbol{Q}[k]\right] = \mathbb{E}\left[\hat{e}(c[k], m[k])|\boldsymbol{Q}[k]\right]$$

Thus, the denominator is common for all terms of inequality (9.22), and it suffices to prove:

$$\mathbb{E}\left[\Delta[k]|\boldsymbol{Q}[k]\right] \leq B + \sum_{n=1}^{N} Q_n[k]\mathbb{E}\left[\lambda_n(\hat{D}(c[k], m[k]) + I[k]) - 1_n[k]|\boldsymbol{Q}[k]\right] \quad (9.23)$$

To this end, by squaring (9.18) and noting that $\max[x, 0]^2 \leq x^2$, we have for each n:

$$\begin{aligned}\frac{1}{2}Q_n[k+1]^2 &\leq \frac{1}{2}(Q_n[k] + \lambda_n(D[k] + I[k]) - 1_n[k])^2 \\ &= \frac{1}{2}Q_n[k]^2 + \frac{1}{2}(\lambda_n(D[k] + I[k]) - 1_n[k])^2 + Q_n[k](\lambda_n(D[k] + I[k]) - 1_n[k])\end{aligned}$$

[2]Indeed, by iterated expectations we have $\mathbb{E}\left[D[k]|\boldsymbol{Q}[k]\right] = \mathbb{E}\left[\mathbb{E}\left[D[k]|(c[k], m[k]), \boldsymbol{Q}[k]\right]|\boldsymbol{Q}[k]\right]$, and $\mathbb{E}\left[D[k]|(c[k], m[k]), \boldsymbol{Q}[k]\right] = \mathbb{E}\left[D[k]|(c[k], m[k])\right]$ because $D[k]$ is conditionally independent of the past given the current $(c[k], m[k])$ used.

Summing the above over $n \in \{1, \ldots, N\}$ and using the definition of $\Delta[k]$ gives:

$$\Delta[k] \leq \frac{1}{2}\sum_{n=1}^{N}(\lambda_n(D[k]+I[k]) - 1_n[k])^2 + \sum_{n=1}^{N} Q_n[k](\lambda_n(D[k]+I[k]) - 1_n[k])$$

Taking conditional expectations given $\boldsymbol{Q}[k]$ and using the bound B proves (9.23). □

The Task Scheduling Algorithm

Our algorithm takes actions every frame to minimize the last two terms on the right-hand-side of the drift-plus-penalty ratio bound (9.22). The only part of these terms that we have control over on frame k (given the observed $\boldsymbol{Q}[k]$) is given below:

$$\frac{\mathbb{E}\left[V\hat{e}(c[k], m[k]) - \sum_{n=1}^{N} Q_n[k]1_n[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c[k], m[k]) + I[k]|\boldsymbol{Q}[k]\right]}$$

Recall from Lemma 1 that minimizing the above ratio of expectations is accomplished over a deterministic choice of $(c[k], m[k], I[k])$. Thus, every frame k we perform the following:

- Observe queues $\boldsymbol{Q}[k] = (Q_1[k], \ldots, Q_N[k])$. Then choose $c[k] \in \{1, \ldots, N\}$, $m[k] \in \mathcal{M}$, and $I[k]$ such that $0 \leq I[k] \leq I_{max}$ to minimize:

$$\frac{V\hat{e}(c[k], m[k]) - Q_{c[k]}[k]}{\hat{D}(c[k], m[k]) + I[k]} \tag{9.24}$$

- Update queues $Q_n[k]$ for each $n \in \{1, \ldots, N\}$ via (9.18), using the $D[k]$, $I[k]$, and $1_n[k]$ values that result from the decisions $c[k]$, $m[k]$, $I[k]$ that minimized (9.24)

Steps to minimize (9.24)

Here we elaborate on how to perform the minimization in (9.24) for each frame k. For each $c \in \{1, \ldots, N\}$ and $m \in \mathcal{M}$, define $idle(c, m)$ as the value of $I[k]$ that minimizes (9.24),

given that we have $(c[k], m[k]) = (c, m)$. It is easy to see that:

$$idle(c,m) = \begin{cases} 0 & \text{if } V\hat{e}(c,m) - Q_c[k] \leq 0 \\ I_{max} & \text{otherwise} \end{cases}$$

Now define $val(c,m)$ by:

$$val(c,m) = \frac{V\hat{e}(c,m) - Q_c[k]}{\hat{D}(c,m) + idle(c,m)}$$

Then we choose $(c[k], m[k])$ as the minimizer of $val(c,m)$ over $c \in \{1,\ldots,N\}$ and $m \in \mathcal{M}$, breaking ties arbitrarily, and choose $I[k] = idle(c[k], m[k])$. Note that this algorithm chooses $I[k] = 0$ or $I[k] = I_{max}$ on every frame k. Nevertheless, it results in a *frame average* $\overline{I}$ that approaches optimality for large V.

9.1.8 Performance of the Task Scheduling Algorithm

For simplicity, the performance theorem is presented in terms of zero initial conditions. It is assumed throughout that the problem (9.3)-(9.6) is feasible, so that it is possible to satisfy the constraints.

Theorem 1. *Suppose $Q_n[0] = 0$ for all $n \in \{1,\ldots,N\}$, and that the problem (9.3)-(9.6) is feasible. Then under the above task scheduling algorithm:*

a) For all frames $K \in \{1,2,3,\ldots\}$ we have:[3]

$$\frac{\overline{e}[K]}{\overline{D}[K] + \overline{I}[K]} \leq power^{opt} + \frac{B}{V(\overline{D}[K] + \overline{I}[K])} \tag{9.25}$$

where B is defined in Lemma 3, $power^{opt}$ is the minimum power solution for the problem (9.3)-(9.6), and $\overline{e}[K]$, $\overline{D}[K]$, $\overline{I}[K]$ are defined by:

$$\overline{e}[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[e[k]\right] \;\;,\;\; \overline{D}[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[D[k]\right] \;\;,\;\; \overline{I}[K] \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[I[k]\right]$$

b) The desired constraints (9.20) and (9.21) are satisfied for all $n \in \{1,\ldots,N\}$. Further,

[3]The right-hand-side in (9.25) can be simplified to $power^{opt} + B/(VD_{min})$, since all frames are at least D_{min} in size.

we have for each frame $K \in \{1, 2, 3, \ldots\}$*:*

$$\frac{\mathbb{E}\left[||\boldsymbol{Q}[k]||\right]}{K} \leq \sqrt{\frac{2(B+V\beta)}{K}} \tag{9.26}$$

where $||\boldsymbol{Q}[k]|| \triangleq \sqrt{\sum_{n=1}^{N} Q_n[k]^2}$ *is the norm of the queue vector (being at least as large as each component* $Q_n[k]$*), and* β *is a constant that satisfies the following for all frames* k*:*

$$\beta \geq \mathbb{E}\left[power^{opt}(D[k]+I[k]) - e[k]\right]$$

Such a constant β *exists because the second moments (and hence first moments) are bounded.*

In the special case of a deterministic system, all expectations of the above theorem can be removed, and the results hold deterministically for all frames K. Theorem 1 indicates that average power can be pushed arbitrarily close to $power^{opt}$, using the V parameter that affects an $O(1/V)$ performance gap given in (9.25). The tradeoff is that V increases the expected size of $\mathbb{E}\left[Q_n[K]\right]/K$ as shown in (9.26), which bounds the expected deviation from the nth constraint during the first K frames (recall (9.19) from Lemma 2). Under a mild additional "Slater-type" assumption that ensures all constraints can be satisfied with "ϵ-slackness," a stronger result on the virtual queues can be shown, namely, that the same algorithm yields queues with average size $O(V)$ [25]. This typically ensures a tighter constraint tradeoff than that given in (9.26). A related improved tradeoff is explored in more detail in Section 9.2.3.

Proof. (Theorem 1 part (a)) Given $\boldsymbol{Q}[k]$ for frame k, our control decisions minimize the last two terms in the right-hand-side of the drift-plus-penalty ratio bound (9.22), and hence:

$$\begin{aligned}\frac{\mathbb{E}\left[\Delta[k]+Ve[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[D[k]+I[k]|\boldsymbol{Q}[k]\right]} \leq\ & \frac{B}{\mathbb{E}\left[D[k]+I[k]|\boldsymbol{Q}[k]\right]} + \frac{\mathbb{E}\left[V\hat{e}(c^*[k],m^*[k])|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c^*[k],m^*[k])+I^*[k]|\boldsymbol{Q}[k]\right]} \\ & + \frac{\sum_{n=1}^{N} Q_n[k]\mathbb{E}\left[\lambda_n(\hat{D}(c^*[k],m^*[k])+I^*[k]) - 1_n^*[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c^*[k],m^*[k])+I^*[k]|\boldsymbol{Q}[k]\right]}\end{aligned} \tag{9.27}$$

where $c^*[k]$, $m^*[k]$, $I^*[k]$, $1_n^*[k]$ are from any alternative (possibly randomized) decisions that can be made on frame k. Now recall the existence of stationary and randomized decisions

that yield (9.15)-(9.16). In particular, these decisions are independent of queue backlog $\boldsymbol{Q}[k]$ and thus yield (from (9.15)):

$$\frac{\mathbb{E}\left[\hat{e}(c^*[k], m^*[k])|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{D}(c^*[k], m^*[k]) + I^*[k]|\boldsymbol{Q}[k]\right]} = \frac{\mathbb{E}\left[\hat{e}(c^*[k], m^*[k])\right]}{\mathbb{E}\left[\hat{D}(c^*[k], m^*[k]) + I^*[k]\right]} = power^{opt}$$

and for all $n \in \{1, \ldots, N\}$ we have (from (9.16)):

$$\mathbb{E}\left[\lambda_n(\hat{D}(c^*[k], m^*[k]) + I^*[k]) - 1_n^*[k]|\boldsymbol{Q}[k]\right] = \mathbb{E}\left[\lambda_n(\hat{D}(c^*[k], m^*[k]) + I^*[k]) - 1_n^*[k]\right] \leq 0$$

Plugging the above into the right-hand-side of (9.27) yields:

$$\frac{\mathbb{E}\left[\Delta[k] + Ve[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]} \leq \frac{B}{\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]} + Vpower^{opt}$$

Rearranging terms gives:

$$\mathbb{E}\left[\Delta[k] + Ve[k]|\boldsymbol{Q}[k]\right] \leq B + Vpower^{opt}\mathbb{E}\left[D[k] + I[k]|\boldsymbol{Q}[k]\right]$$

Taking expectations of the above (with respect to the random $\boldsymbol{Q}[k]$) and using the law of iterated expectations gives:

$$\mathbb{E}\left[\Delta[k] + Ve[k]\right] \leq B + Vpower^{opt}\mathbb{E}\left[D[k] + I[k]\right] \tag{9.28}$$

The above holds for all $k \in \{0, 1, 2, \ldots\}$. Fixing a positive integer K and summing (9.28) over $k \in \{0, 1, \ldots, K-1\}$ yields, by the definition $\Delta[k] = L[k+1] - L[k]$:

$$\mathbb{E}\left[L[K] - L[0]\right] + V\sum_{k=0}^{K-1}\mathbb{E}\left[e[k]\right] \leq BK + Vpower^{opt}\sum_{k=0}^{K-1}\mathbb{E}\left[D[k] + I[k]\right]$$

Noting that $L[0] = 0$ and $L[K] \geq 0$ and using the definitions of $\overline{e}[K]$, $\overline{D}[K]$, $\overline{I}[K]$ yields:

$$VK\overline{e}[K] \leq BK + VKpower^{opt}(\overline{D}[K] + \overline{I}[K])$$

Rearranging terms yields the result of part (a). □

Proof. (Theorem 1 part (b)) To prove part (b), note from (9.28) we have:

$$\mathbb{E}\left[\Delta[k]\right] \leq B + V\mathbb{E}\left[power^{opt}(D[k] + I[k]) - e[k]\right] \leq B + V\beta$$

Summing the above over $k \in \{0, 1, \ldots, K-1\}$ gives:

$$\mathbb{E}\left[L[K]\right] - \mathbb{E}\left[L[0]\right] \leq (B + V\beta)K$$

Using the definition of $L[K]$ and noting that $L[0] = 0$ gives:

$$\sum_{l=1}^{K} \mathbb{E}\left[Q_n[K]^2\right] \leq 2(B + V\beta)K$$

Thus, we have $\mathbb{E}\left[||\boldsymbol{Q}[K]||^2\right] \leq 2(B + V\beta)K$. Jensen's inequality for $f(x) = x^2$ ensures $\mathbb{E}\left[||\boldsymbol{Q}[k]||\right]^2 \leq \mathbb{E}\left[||\boldsymbol{Q}[k]||^2\right]$, and so for all positive integers K:

$$\mathbb{E}\left[||\boldsymbol{Q}[k]||\right]^2 \leq 2(B + V\beta)K \tag{9.29}$$

Taking a square root of both sides of (9.29) and dividing by K proves (9.26). From (9.26) we have for each $n \in \{1, \ldots, N\}$:

$$\lim_{K\to\infty} \frac{\mathbb{E}\left[Q_n[k]\right]}{K} \leq \lim_{K\to\infty} \frac{\mathbb{E}\left[||\boldsymbol{Q}[K]||\right]}{K} \leq \lim_{K\to\infty} \frac{\sqrt{2(B + V\beta)}}{\sqrt{K}} = 0$$

and hence by Lemma 2 we know constraint (9.21) holds. Further, in [27] it is shown that (9.29) together with the fact that second moments of queue changes are bounded implies $\lim_{k\to\infty} Q_n[k]/k = 0$ with probability 1. Thus, (9.20) holds. □

9.1.9 Simulation

We first simulate the task scheduling algorithm for the simple deterministic system with one class and one constraint, as described in Section 9.1.4. The $\hat{e}(m)$ and $\hat{D}(m)$ functions are defined in (9.8)-(9.9), and the goal is to minimize average power subject to a processing rate constraint $1/(\overline{D} + \overline{I}) \geq 0.2$. We already know the optimal power is $power^{opt} = 7/15 \approx$

0.466667. We expect the algorithm to approach this optimal power as V is increased, and to approach the desired behavior of using $I[k] = 0$ for all k, meeting the constraint with equality, and using $m[k] = 1$ for 1/3 of the frames. This is indeed what happens, although in this simple case the algorithm seems insensitive to the V parameter and locks into a desirable periodic schedule even for very low (but positive) V values. Using $V = 1$ and one million frames, the algorithm gets average power 0.466661, uses $m[k] = 1$ a fraction of time 0.333340, has average idle time $\overline{I} = 0.000010$, and yields a processing rate 0.199999 (almost exactly equal to the desired constraint of 0.2). Increasing V yields similar performance. The constraint is still satisfied when we decrease the value of V, but average power degrades (being 0.526316 for $V = 0$).

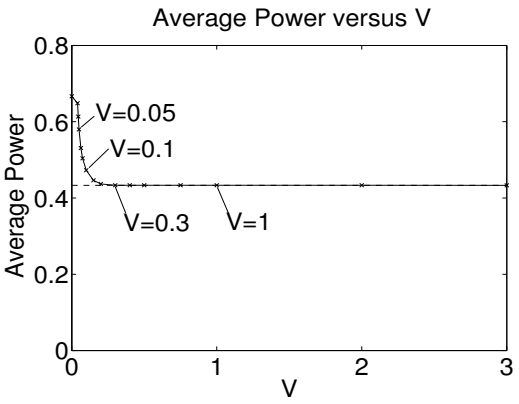

Figure 9.3: Average power versus V.

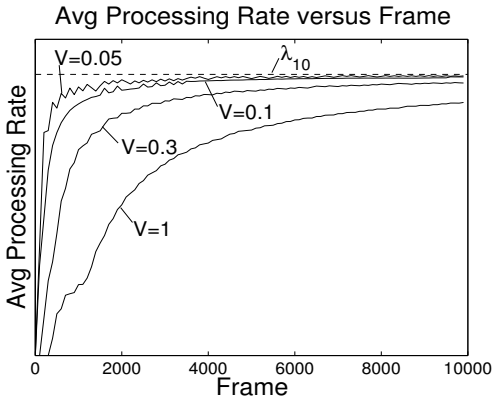

Figure 9.4: Processing Rate $\overline{1}_{10}/\overline{T}$ versus frame index.

We next consider a system with 10 classes of tasks and two processing modes. The energy and delay characteristics for each class $i \in \{1, \ldots, 10\}$ and mode $m \in \{1, 2\}$ are:

$$\text{Mode 1:} \quad (\hat{e}(i,1), \hat{D}(i,1)) = (1i, 5i) \tag{9.30}$$

$$\text{Mode 2:} \quad (\hat{e}(i,2), \hat{D}(i,2)) = (2i, 3i) \tag{9.31}$$

so that mode 1 uses less energy but takes longer than mode 2, and the computational requirements for each class increase with $i \in \{1, \ldots, 10\}$. We assume desired rates are given by $\lambda_i = \rho/(30i)$ for $i \in \{1, \ldots, 10\}$, for some positive value ρ. The problem is feasible whenever $\rho \leq 1$. We use $\rho = 0.8$ and run the simulation for 10 million frames. Fig. 9.3 shows the resulting average power as V is varied between 0 and 3, which converges to near

optimal after $V = 0.3$. All 10 processing rate constraints are met within 5 decimal points of accuracy after the 10 million frames. An illustration of how convergence time is affected by the V parameter is shown in Fig. 9.4, which illustrates the average processing rate $\overline{1}_{10}/\overline{T}$ for class 10, as compared to the desired constraint $\lambda_{10} = \rho/300$. It is seen, for example, that convergence is faster for $V = 0.05$ than for $V = 1$. Convergence times can be improved using non-zero initial queue backlog and the theory of *place holder backlog* in [25], although we omit this topic for brevity.

9.2 Optimization with General Attributes

This section generalizes the problem to allow time average optimization for abstract *attributes*. Consider again a frame-based system with frame index $k \in \{0, 1, 2, \ldots\}$. Every frame k, the controller makes a *control action* $\alpha[k]$, chosen within an abstract set $\mathcal{A}$ of allowable actions. The action $\alpha[k]$ affects the *frame size* $T[k]$ and an *attribute vector* $\boldsymbol{y}[k] = (y_0[k], y_1[k], \ldots, y_L[k])$. Specifically, assume these are random functions that are conditionally independent of the past given the current $\alpha[k]$ decision, with mean values given by functions $\hat{T}(\alpha)$ and $\hat{y}_l(\alpha)$ for all $\alpha \in \mathcal{A}$:

$$\hat{T}(\alpha) = \mathbb{E}\left[T[k]|\alpha[k] = \alpha\right] \ , \ \hat{y}_l(\alpha) = \mathbb{E}\left[y_l[k]|\alpha[k] = \alpha\right]$$

Similar to the previous section, it is assumed there is a minimum frame size $T_{min} > 0$ such that $T[k] \geq T_{min}$ for all k, and that second moments are bounded by a constant σ^2, regardless of the policy $\alpha[k]$. The joint distribution of $(T[k], y_0[k], y_1[k], \ldots, y_L[k])$ is otherwise arbitrary.

Define frame averages $\overline{T}$ and $\overline{y}_l$ by:

$$\overline{T} = \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} T[k] \ , \ \overline{y}_l = \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} y_l[k]$$

As discussed in Section 9.1.2, the value $\overline{y}_l/\overline{T}$ represents the *time average* associated with

attribute $y_l[k]$. The general problem is then:

$$\text{Minimize:} \quad \overline{y}_0/\overline{T} \tag{9.32}$$

$$\text{Subject to:} \quad \overline{y}_l/\overline{T} \leq c_l \quad \forall l \in \{1, \ldots, L\} \tag{9.33}$$

$$\alpha[k] \in \mathcal{A} \quad \forall k \in \{0, 1, 2, \ldots\} \tag{9.34}$$

where $c_1, \ldots, c_L$ are given constants that specify the desired time average constraints.

9.2.1 Mapping to the Task Scheduling Problem

To illustrate the generality of this framework, this subsection uses the new notation to exactly represent the task scheduling problem from Section 9.1. For that problem, one can define the control action $\alpha[k]$ to have the form $\alpha[k] = (c[k], m[k], I[k])$, and the action space $\mathcal{A}$ is then the set of all (c, m, I) such that $c \in \{1, \ldots, N\}$, $m \in \mathcal{M}$, and $0 \leq I \leq I_{max}$.

The frame size is $T[k] = D[k] + I[k]$, and $\hat{T}(\alpha[k])$ is given by:

$$\hat{T}(\alpha[k]) = \hat{D}(c[k], m[k]) + I[k]$$

We then define $y_0[k]$ as the energy expended in frame k, so that $y_0[k] = e[k]$ and $\hat{y}_0(\alpha[k]) = \hat{e}(c[k], m[k])$. There are N constraints, so define $L = N$. To express the desired constraints $\overline{1}_n/\overline{T} \geq \lambda_n$ in the form $\overline{y}_n/\overline{T} \leq c_n$, one can define $y_n[k] = -1_n[k]$ and $c_n = -\lambda_n$ for each $n \in \{1, \ldots, N\}$, and $\hat{y}_n(\alpha[k]) = -1_n[k]$. Alternatively, one could define $y_n[k] = \lambda_n T[k] - 1_n[k]$ and enforce the constraint $\overline{y}_n \leq 0$ for all $n \in \{1, \ldots, N\}$.

This general setup provides more flexibility. For example, suppose the idle state does not use 0 energy, but operates at a low power p_{idle} and expends total energy $p_{idle}I[k]$ on frame k. Then total energy for frame k can be defined as $y_0[k] = e[k] + p_{idle}I[k]$, where $e[k]$ is the energy spent in the busy period. The setup can also handle systems with multiple idle mode options, each providing a different energy savings but incurring a different wakeup time.

9.2.2 The General Algorithm

The algorithm for solving the general problem (9.32)-(9.34) is described below. Each constraint $\overline{y}_l \leq c_l\overline{T}$ in (9.33) is treated using a virtual queue $Q_l[k]$ with update:

$$Q_l[k+1] = \max[Q_l[k] + y_l[k] - c_l T[k], 0] \ \forall l \in \{1, \ldots, L\} \tag{9.35}$$

Defining $L[k]$ and $\Delta[k]$ as before (in Section 9.1.7) leads to the following bound on the drift-plus-penalty ratio, which can be proven in a manner similar to Lemma 3:

$$\frac{\mathbb{E}\left[\Delta[k] + Vy_0[k]|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[T[k]|\boldsymbol{Q}[k]\right]} \leq \frac{B}{\mathbb{E}\left[T[k]|\boldsymbol{Q}[k]\right]} + \frac{\mathbb{E}\left[V\hat{y}_0(\alpha[k]) + \sum_{l=1}^{L} Q_l[k]\hat{y}_l(\alpha[k])|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{T}(\alpha[k])|\boldsymbol{Q}[k]\right]} \tag{9.36}$$

where B is a constant that satisfies the following for all $\boldsymbol{Q}[k]$ and all possible actions $\alpha[k]$:

$$B \geq \frac{1}{2}\sum_{l=1}^{L} \mathbb{E}\left[(y_l[k] - c_l T[k])^2|\boldsymbol{Q}[k]\right]$$

Every frame, the controller observes queues $\boldsymbol{Q}[k]$ and takes an action $\alpha[k] \in \mathcal{A}$ that minimizes the second term on the right-hand-side of (9.36). We know from Lemma 1 that minimizing the ratio of expectations is accomplished by a deterministic selection of $\alpha[k] \in \mathcal{A}$. The resulting algorithm is:

- Observe $\boldsymbol{Q}[k]$ and choose $\alpha[k] \in \mathcal{A}$ to minimize (breaking ties arbitrarily):

$$\frac{V\hat{y}_0(\alpha[k]) + \sum_{l=1}^{L} Q_l[k]\hat{y}_l(\alpha[k])}{\hat{T}(\alpha[k])} \tag{9.37}$$

- Update virtual queues $Q_l[k]$ for each $l \in \{1, \ldots, L\}$ via (9.35).

One subtlety is that the expression (9.37) may not have an achievable minimum over the general (possibly infinite) set $\mathcal{A}$ (for example, the infimum of the function $f(x) = x$ over the open interval $0 < x < 1$ is not achievable over that interval). This is no problem: Our

algorithm in fact works for any *approximate* minimum that is an additive constant C away from the exact infimum every frame k (for any arbitrarily large constant $C \geq 0$). This effectively changes the "B" constant in our performance bounds to a new constant "$B + C$" [25]. Let $ratio^{opt}$ represent the optimal ratio of $\overline{y}_0/\overline{T}$ for the problem (9.32)-(9.34). As before, it can be shown that if the problem is feasible (so that there exists an algorithm that can achieve the constraints (9.33)-(9.34)), then any C-additive approximation of the above algorithm satisfies all desired constraints and yields $\overline{y}_0/\overline{T} \leq ratio^{opt} + O(1/V)$, which can be pushed arbitrarily close to $ratio^{opt}$ as V is increased, with the same tradeoff in the queue sizes (and hence convergence times) with V. The proof of this is similar to that of Theorem 1, and is omitted for brevity (see [25][30] for the full proof).

9.2.3 Random Task Arrivals and Flow Control

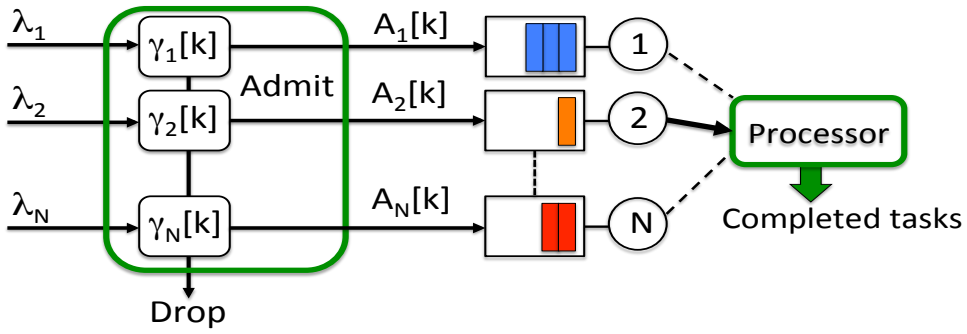

Figure 9.5: A task processing system with random arrivals and flow control.

Again consider a system with N classes of tasks, as in Section 9.1. Each frame k again has a busy period of duration $D[k]$ and an idle period of duration $I[k]$ as in Fig. 9.1. However, rather than always having tasks available for processing, this subsection assumes tasks arrive *randomly* with rates $(\lambda_1, \ldots, \lambda_N)$, where λ_n is the rate of task arrivals per unit time (see Fig. 9.5). At the beginning of each busy period, the controller chooses a variable $c[k]$ that specifies which type of task is performed. However, $c[k]$ can now take values in the set $\{0, 1, \ldots, N\}$, where $c[k] = 0$ is a *null* choice that selects no task on frame k. If $c[k] = 0$, the busy period has some positive size D_0 and may spend a small amount of energy to power the electronics, but does not process any task. The mode selection variable $m[k]$ takes values in the same set $\mathcal{M}$ as before. The idle time variable $I[k]$ is again chosen in the interval $[0, I_{max}]$.

Further, for each $n \in \{1, 2, \ldots, N\}$ we introduce *flow control variables* $\gamma_n[k]$, chosen in the interval $0 \leq \gamma_n[k] \leq 1$. The variable $\gamma_n[k]$ represents the probability of admitting each new randomly arriving task of class n on frame k (see Fig. 9.5). This enables the system to drop tasks if the raw arrival rates $(\lambda_1, \ldots, \lambda_N)$ cannot be supported. Let $\boldsymbol{\gamma}[k] = (\gamma_1[k], \ldots, \gamma_N[k])$ be the vector of these variables.

We thus have $\alpha[k] = (c[k], m[k], I[k], \boldsymbol{\gamma}[k])$, with action space $\mathcal{A}$ being the set of all $(c, m, I, \boldsymbol{\gamma})$ such that $c \in \{0, 1, \ldots, N\}$, $m \in \mathcal{M}$, $0 \leq I \leq I_{max}$, and $0 \leq \gamma_n \leq 1$ for $n \in \{1, \ldots, N\}$. Define $e[k]$ and $D[k]$ as the energy and busy period duration for frame k. Assume $e[k]$ depends only on $(c[k], m[k], I[k])$, and $D[k]$ depends only on $(c[k], m[k])$, with averages given by functions $\hat{e}(c[k], m[k], I[k])$ and $\hat{D}(c[k], m[k])$:

$$\hat{e}(c[k], m[k], I[k]) = \mathbb{E}\left[e[k]|c[k], m[k], I[k]\right] \quad , \quad \hat{D}(c[k], m[k]) = \mathbb{E}\left[D[k]|c[k], m[k]\right]$$

Finally, for each $n \in \{1, \ldots, N\}$, define $A_n[k]$ as the random number of new arrivals admitted on frame k, which depends on the total frame size $D[k] + I[k]$ and the admission probability $\gamma_n[k]$. Formally, assume the arrival vector $(A_1[k], \ldots, A_N[k])$ is conditionally independent of the past given the current $\alpha[k]$ used, with expectations:

$$\mathbb{E}\left[A_n[k]|\alpha[k]\right] = \lambda_n \gamma_n[k][\hat{D}(c[k], m[k]) + I[k]] \tag{9.38}$$

The assumption on independence of the past holds whenever arrivals are independent and Poisson, or when all frame sizes are an integer number of fixed size slots, and arrivals are independent and identically distributed (i.i.d.) over slots with some general distribution.

We seek to maximize a weighted sum of admission rates subject to supporting all of the admitted tasks, and to maintaining average power to within a given positive constant P_{av}:

$$\text{Maximize:} \quad \frac{\sum_{n=1}^{N} w_n \overline{A}_n}{\overline{D} + \overline{I}} \tag{9.39}$$

$$\text{Subject to:} \quad \overline{A}_n/(\overline{D} + \overline{I}) \leq \overline{1}_n/(\overline{D} + \overline{I}) \quad \forall n \in \{1, \ldots, N\} \tag{9.40}$$

$$\frac{\overline{e}}{\overline{D} + \overline{I}} \leq P_{av} \tag{9.41}$$

$$\alpha[k] \in \mathcal{A} \quad \forall k \in \{0, 1, 2, \ldots\} \tag{9.42}$$

where $(w_1, \ldots, w_N)$ are a collection of positive weights that prioritize the different classes in the optimization objective.

We thus have $L = N + 1$ constraints. To treat this problem, define $T[k] = D[k] + I[k]$, so that $\hat{T}(\alpha[k]) = \hat{D}(c[k], m[k]) + I[k]$. Further define $y_0[k]$, $y_n[k]$ for $n \in \{1, \ldots, N\}$, and $x[k]$ as:

$$\begin{aligned} y_0[k] = -\sum_{n=1}^{N} w_n A_n[k] \quad &\Longrightarrow \quad \hat{y}_0(\alpha[k]) = -\sum_{n=1}^{N} w_n \lambda_n \gamma_n[k][\hat{D}(c[k], m[k]) + I[k]] \\ y_n[k] = A_n[k] - 1_n[k] \quad &\Longrightarrow \quad \hat{y}_n(\alpha[k]) = \lambda_n \gamma_n[k][\hat{D}(c[k], m[k]) + I[k]] - 1_n[k] \\ x[k] = e[k] - [D[k] + I[k]]P_{av} \quad &\Longrightarrow \quad \hat{x}(\alpha[k]) = \hat{e}(c[k], m[k], I[k]) - [\hat{D}(c[k], m[k]) + I[k]]P_{av} \end{aligned}$$

Then:

- Minimizing $\overline{y}_0/\overline{T}$ is equivalent to (9.39).
- The constraints $\overline{y}_n \leq 0$ for $n \in \{1, \ldots, N\}$ are equivalent to (9.40).
- The constraint $\overline{x} \leq 0$ is equivalent to (9.41).

Note that the above problem does not specify any explicit queueing for the randomly arriving tasks. The algorithm will in fact construct explicit queues (so that the virtual queues can be viewed as actual queues). Note also that the constraint $\alpha[k] \in \mathcal{A}$ does not allow restrictions on actions based on the queue state, such as when the queue is empty or not. Thus, in principle, we allow the possibility of "processing" a task of class n even when there is no such task available. In this case, we assume this processing is still costly, in that it incurs time equal to $\hat{D}(c[k], m[k])$ and energy equal to $\hat{e}(c[k], m[k], I[k])$. Our algorithm will naturally learn to avoid the inefficiencies associated with such actions.

The Dynamic Algorithm for Random Task Arrivals

To enforce the constraints $\overline{y}_n \leq 0$ for each $n \in \{1, \ldots, N\}$, define queue $Q_n[k]$ with update:

$$Q_n[k+1] = \max[Q_n[k] + A_n[k] - 1_n[k], 0] \tag{9.43}$$

To enforce $\overline{x} \leq 0$, define a virtual queue $Z[k]$ with update:

$$Z[k+1] = \max[Z[k] + e[k] - [D[k] + I[k]]P_{av}, 0] \tag{9.44}$$

It can be seen that the queue update (9.43) is the same as that of an *actual queue* for class n tasks, with random task arrivals $A_n[k]$ and task service $1_n[k]$. The minimization of (9.37) then becomes the following: Every frame k, observe queues $\boldsymbol{Q}[k]$ and $Z[k]$. Then choose $c[k] \in \{0, 1, \ldots, N\}$, $m[k] \in \mathcal{M}$, $I[k] \in [0, I_{max}]$, and $\gamma_n[k] \in [0, 1]$ for all $n \in \{1, \ldots, N\}$ to minimize:

$$\begin{aligned} &\frac{-V\sum_{n=1}^{N} w_n\lambda_n\gamma_n[k][\hat{D}(c[k], m[k]) + I[k]]}{\hat{D}(c[k], m[k]) + I[k]} \\ &+\frac{\sum_{n=1}^{N} Q_n[k](\lambda_n\gamma_n[k][\hat{D}(c[k], m[k]) + I[k]] - 1_n[k])}{\hat{D}(c[k], m[k]) + I[k]} \\ &+\frac{Z[k](\hat{e}(c[k], m[k], I[k]) - [\hat{D}(c[k], m[k]) + I[k]]P_{av})}{\hat{D}(c[k], m[k]) + I[k]} \end{aligned} \tag{9.45}$$

After a simplifying cancellation of terms, it is easy to see that the $\gamma_n[k]$ decisions can be separated from all other decisions (see Exercise 2). The resulting algorithm then observes queues $\boldsymbol{Q}[k]$ and $Z[k]$ every frame k and performs the following:

- (Flow Control) For each $n \in \{1, \ldots, N\}$, choose $\gamma_n[k]$ as:

$$\gamma_n[k] = \begin{cases} 1 & \text{if } Q_n[k] \leq Vw_n \\ 0 & \text{otherwise} \end{cases} \tag{9.46}$$

- (Task Scheduling) Choose $c[k] \in \{0, 1, \ldots, N\}$, $m[k] \in \mathcal{M}$, $I[k] \in [0, I_{max}]$ to minimize:

$$\frac{Z[k]\hat{e}(c[k], m[k], I[k]) - \sum_{n=1}^{N} Q_n[k]1_n[k]}{\hat{D}(c[k], m[k]) + I[k]} \tag{9.47}$$

- (Queue Update) Update $Q_n[k]$ for each $n \in \{1, \ldots, N\}$ by (9.43), and update $Z[k]$ by (9.44).

The minimization problem (9.47) is similar to (9.24), and can be carried out in the same manner as discussed in Section 9.1.7. A key observation about the above algorithm is that *it does not require knowledge of the arrival rates* $(\lambda_1, \ldots, \lambda_N)$. Indeed, the λ_n terms cancel out of the minimization, so that the flow control variables $\gamma_n[k]$ in (9.46) make "bang-bang" decisions that admit all newly arriving tasks of class n on frame k if $Q_n[k] \leq Vw_n$, and admit none otherwise. This property makes the algorithm naturally adaptive to situations when the arrival rates change, as shown in the simulations of Section 9.2.4.

Note that if $\hat{e}(0, m, I) < \hat{e}(c, m, I)$ for all $c \in \{1, \ldots, N\}$, $m \in \mathcal{M}$, $I \in [0, I_{max}]$, so that the energy associated with processing *no task* is less than the energy of processing any class $c \neq 0$, then the minimization in (9.47) will never select a class c such that $Q_c[k] = 0$. That is, the algorithm naturally will never select a class for which no task is available.

Deterministic Queue Bounds and Constraint Violation Bounds

In addition to satisfying the desired constraints and achieving a weighted sum of admitted rates that is within $O(1/V)$ of optimality, the flow control structure of the task scheduling algorithm admits *deterministic queue bounds*. Specifically, assume all frame sizes are bounded by a constant T_{max}, and that the raw number of class n arrivals per frame (before admission control) is at most $A_{n,max}$. By the flow control policy (9.46), new arrivals of class n are only admitted if $Q_n[k] \leq Vw_n$. Thus, assuming that $Q_n[0] \leq Vw_n + A_{n,max}$, we must have $Q_n[k] \leq Vw_n + A_{n,max}$ for all frames $k \in \{0, 1, 2, \ldots\}$. This specifies a worst-case queue backlog that is $O(V)$, which establishes an explicit $[O(1/V), O(V)]$ performance-backlog tradeoff that is superior to that given in (9.26).

With mild additional structure on the $\hat{e}(c, m, I)$ function, the deterministic bounds on queues $Q_n[k]$ lead to a deterministic bound on $Z[k]$, so that one can compute a value Z_{max}, of size $O(V)$, such that $Z[k] \leq Z_{max}$ for all k. This is explored in Exercise 3 (see also [29]).

9.2.4 Simulations and Adaptiveness of Random Task Scheduling

Here we simulate the dynamic task scheduling and flow control algorithm (9.46)-(9.47), using the 10-class system model defined in Section 9.1.9 with $\hat{e}(c, m)$ functions given in (9.30)-(9.31). For consistency with that model, we remove the $c[k] = 0$ option, so that the decision (9.31) chooses $c[k] = 1, m[k] = 1$, incurring one unit of energy, in case no tasks are available. Arrivals are from independent Bernoulli processes with rates $\lambda_i = \rho/(30i)$ for each class $i \in \{1, \ldots, 10\}$, with $\rho = 0.8$. We use weights $w_n = 1$ for all n, so that the objective is to maximize total throughput, and $P_{av} = 0.5$, which we know is feasible from results in Fig. 9.3 of Section 9.1.9. Thus, we expect the algorithm to learn to admit everything, so that the admission rate approaches the total arrival rate as V is increased. We simulate for 10 million frames, using V in the interval from 0 to 200. Results are shown in Figs. 9.6 and 9.7. Fig 9.6 shows the algorithm learns to admit everything for large V (100 or above), and Fig. 9.7 plots the resulting average queue size (in number of tasks) per queue, together with the deterministic bound $Q_n[k] \leq V + 60$ (where $A_{n,max} = 60$ in this case because there is at most one arrival per slot, and the largest possible frame is $D_{max} + I_{max} = 50 + 10 = 60$). The average power constraint was satisfied (with slackness) for all cases. The average queue size in Fig. 9.7 grows linearly with V until $V = 100$, when it saturates by admitting everything. The saturation value is the average queue size associated with admitting the raw arrival rates directly.

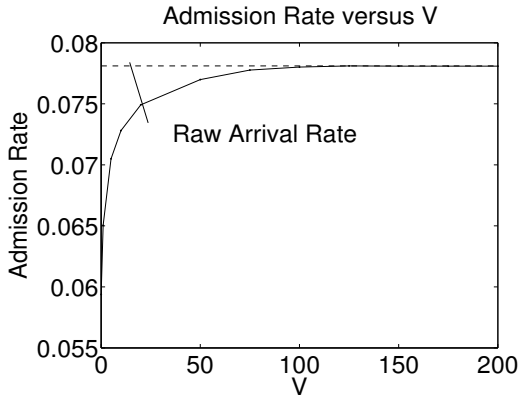

Figure 9.6: Total admission rate versus V.

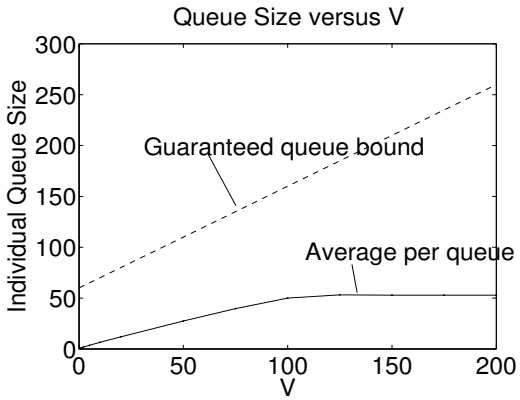

Figure 9.7: Average queue size per queue, and the deterministic bound from Section 9.2.3.

We now illustrate that the algorithm is robust to abrupt rate changes. We consider the same system as before, run over 10 million frames with $V = 100$. However, we break the simulation timeline into three equal size phases. During the first and third phase, we use arrival rates $\lambda_i = \rho_1/(30i)$ for $i \in \{1, \ldots, 10\}$, where $\rho_1 = 0.8$ as in the previous experiment. Then, during the second (middle) phase, we double the rates to $\lambda_i = \rho_2/(30i)$, where $\rho_2 = 1.6$. Because $\rho_2 > 1$, these rates are *infeasible* and the algorithm must learn to optimally drop tasks so as to maximize the admission rate subject to the power constraint. Recall that the algorithm is unaware of the arrival rates and must adapt to the existing system conditions. The results are shown in Figs. 9.8 and 9.9. Fig. 9.8 shows a moving average admission rate versus time. During the first and third phases of the simulation, we have $\rho_1 = 0.8$ and the admitted rates are close to the raw arrival rate (shown as the lower dashed horizontal line). During the middle interval (with $\rho_2 = 1.6$), the algorithm quickly adapts to the increased arrivals and yields admitted rates that are close to those that should be used in a system with loading $\rho_2 = 1.6$ always (shown as the higher dashed horizontal line). Fig. 9.9 plots the corresponding moving average queue backlog per queue. The lower dashed horizontal line indicates the value of average backlog that would be achieved in a system with loading $\rho_1 = 0.8$ always. Also shown is the deterministic queue bound $V + 60 = 160$, which holds for all frames, regardless of the raw arrival rates.

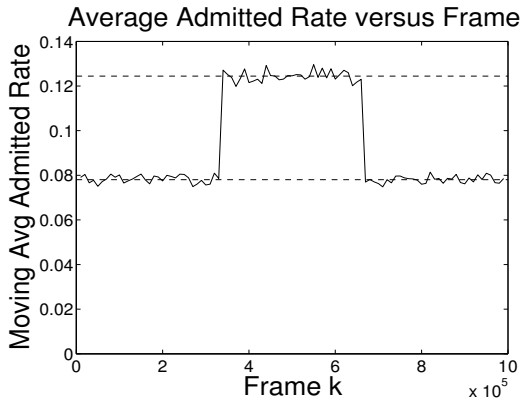

Figure 9.8: Moving average admission rate versus frame index k for the system with abrupt rate changes.

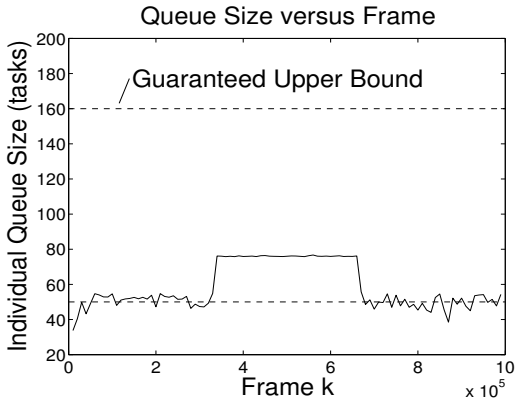

Figure 9.9: Individual queue size (average and guaranteed bound) versus frame index k for the system with abrupt rate changes.

9.2.5 Task Scheduling: Extensions and Further Reading

This section considered minimizing time averages subject to time average constraints. Extended techniques for minimizing convex functions of time average vectors (or maximizing concave functions) subject to similar constraints are treated in the renewal optimization theory of [25][30]. This is useful for extending the flow control problem (9.39)-(9.42) to optimize a sum of concave utility functions $\phi_n(x)$ of the time average admission rates for each class:

$$\sum_{n=1}^{N} \phi_n\left(\overline{A}_n/(\overline{D}+\overline{I})\right)$$

Using logarithmic functions $\phi_n(x)$ leads to *proportional fairness* [15], while other fairness properties are achieved with other concave functions [23][37][41]. Related utility optimization for particular classes of wireless systems with fixed size frames of T slots, and with probabilistic reception every slot, is treated using a different technique in [11][10]. Related work on utility optimal scheduling for single-slot data networks is treated using convex optimization in [16][22][43][21], stochastic "primal-dual" algorithms in [17][1][36][4][20], and stochastic "dual" algorithms in [24][32][5][8][34].

For the problem with random arrivals, queueing delay can often be improved by changing the constraint $\overline{A}_n \leq \overline{1}_n$ of (9.40) to $\overline{A}_n + \epsilon \leq \overline{1}_n$, for some $\epsilon > 0$. This specifies that the output processing rate should be ϵ larger than the arrival rate. However, unlike the case $\epsilon = 0$, such a constraint yields a queue $Q_n[k]$ that contains some "fake" tasks, and can lead to decisions that serve the queue when no actual tasks are available. This can be overcome using the theory of *ϵ-persistent service queues* in [33][26].

Note that our flow control algorithm rewards admission of new data on each frame. This fits the general framework of this section, where attributes $y_l[k]$ are (possibly random) functions of the control action $\alpha[k] \in \mathcal{A}$. One might attempt to solve the problem by defining a reward upon *completion* of service. This does not fit our framework: That is because the algorithm might "complete" a service in a queue that is empty, simply to accumulate the "fake" reward. One could eliminate fake rewards by an augmented model that allows rewards $\hat{y}_l(\alpha[k], \boldsymbol{Q}[k])$ and/or action spaces $\mathcal{A}(\boldsymbol{Q}[k])$ that depend on the current backlog,

although this creates a much more complex Markov decision model that is easily avoided by defining rewards at admission. However, there are some problems that require such a reward structure, such as problems of stock market trading where prior ownership of a particular stock is required to reap the benefits of selling at an observed desirable price. These problems can be treated with a modified Lyapunov function of the form $L[k] = \sum_{n=1}^{N}(Q_n[k] - \theta)^2$, which pushes backlog towards a non-empty state θ. This approach is used for stock trading [28], inventory control [31], energy harvesting [13], and smart-grid energy management [40]. The first algorithms for optimal energy harvesting used a related technique [7], and related problems in processing networks that assemble components by combining different sub-components are treated in [14][12].

9.2.6 Exercises for Section 9.2

Exercise 1. *Consider a system with N classes, each of which always has new tasks available. Every frame k we choose a class $c[k] \in \{1, \ldots, N\}$, and select a single task of that class for processing. We can process using mode $m[k] \in \mathcal{M}$. The result yields frame size $\hat{T}(c[k], m[k])$, energy $\hat{e}(c[k], m[k])$, and* processing quality *$\hat{q}(c[k], m[k])$. Design an algorithm that selects $c[k]$ and $m[k]$ every frame k to maximize $\overline{q}/\overline{T}$ subject to an average power constraint $\overline{e}/\overline{T} \leq P_{av}$ and to a processing rate constraint of at least λ_n for each class n.*

Exercise 2. *Show that minimization of* (9.45) *leads to the separable flow control and task scheduling algorithm of* (9.46)-(9.47)*.*

Exercise 3. *Consider the flow control and task scheduling algorithm* (9.46)-(9.47)*, and recall that $Q_n[k] \leq Q_{max}$ for all $n \in \{1, \ldots, N\}$ and all frames k, where $Q_{max} \triangleq \max_{n\in\{1,\ldots,N\}}[Vw_n + A_{n,max}]$. Suppose there is a constant $e_{min} > 0$ such that $\hat{e}(c, m, I) \geq e_{min}$ for all $c \neq 0$, and that $\hat{e}(0, m, I) = 0$ for all $m \in \mathcal{M}$, $I \in [0, I_{max}]$.*

a) Show that the minimization of (9.47) *chooses $c[k] = 0$ whenever $Z[k] > Q_{max}/e_{min}$.*

b) Suppose there is a constant e_{max} such that $e[k] \leq e_{max}$ for all k. Conclude that $Z[k] \leq Z_{max} \triangleq Q_{max}/e_{min} + e_{max}$ for all frames k.

c) Use the queueing equation (9.44) *to show that over any sequence of K frames* $\{j, j+1, \ldots, j+K-1\}$ *(starting at some frame j), the total energy usage satisfies:*

$$\sum_{k=j}^{j+K-1} e[k] \leq P_{av} \sum_{k=j}^{j+K-1} T[k] + Z_{max}$$

where $T[k]$ *is the size of frame k. Hint: Make an argument similar to the proof of Lemma 2.*

Exercise 4. *Consider the same general attributes* $y_l[k]$ *of Section 9.2, with* $\hat{y}_l(\alpha[k])$ *for* $\alpha[k] \in \mathcal{A}$*. State the algorithm for solving the problem of minimizing* $\overline{y}_0$ *subject to* $\overline{y}_l/\overline{T} \leq c_l$*. Hint: Define appropriate attributes* $x_l[k]$ *on a system with an "effective" frame size of 1 every frame, and minimize* $\overline{x}_0/1$ *subject to* $\overline{x}_l/1 \leq 0$ *for* $l \in \{1, \ldots, L\}$*.*

Exercise 5. *Modify the task scheduling algorithm of Section 9.1.7 to allow the controller to serve more than one task per frame. Specifically, every frame k the controller chooses a* service action $s[k]$ *from a set* $\mathcal{S}$ *of possible actions. Each service action* $s \in \mathcal{S}$ *determines a* clearance vector $\boldsymbol{c}(s) = (c_1(s), \ldots, c_N(s))$*, where* $c_i(s)$ *is the number of tasks of type i served if action s is used on a given frame. It also incurs a delay* $\hat{D}(s)$ *and energy* $\hat{e}(s)$*.*

Exercise 6. *Consider the linear fractional problem of finding a vector* $(x_1, \ldots, x_M)$ *to minimize* $(a_0 + \sum_{i=1}^M a_i x_i)/(b_0 + \sum_{i=1}^M b_i x_i)$ *subject to* $\sum_{i=1}^M c_{il} x_i \leq d_l$ *for* $l \in \{1, \ldots, L\}$ *and* $0 \leq x_i \leq 1$ *for all* $i \in \{1, \ldots, M\}$*. Assume constants* a_i*,* b_i*,* c_{il}*,* d_l *are given, that* $b_0 > 0$*, and that* $b_i \geq 0$ *for all* $i \in \{1, \ldots, M\}$*. Treat this as a time average problem with action* $\alpha[k] = (x_1[k], \ldots, x_M[k])$*, action space* $\mathcal{A} = \{(x_1, \ldots, x_M) | 0 \leq x_i \leq 1 \ \forall i \in \{1, \ldots, M\}\}$*, frame size* $T[k] = b_0 + \sum_{i=1}^M b_i x_i[k]$*, and where we seek to minimize* $(a_0 + \sum_{i=1}^M a_i \overline{x}_i)/(b_0 + \sum_{i=1}^M b_i \overline{x}_i)$ *subject to* $\sum_{i=1}^M c_{il} \overline{x}_i \leq d_l$ *for all* $l \in \{1, \ldots, L\}$*.*

a) State the drift-plus-penalty ratio algorithm (9.37) *for this, and conclude that:*

$$\lim_{K\to\infty} \frac{a_0 + \sum_{i=1}^M a_i \overline{x}_i[K]}{b_0 + \sum_{i=1}^M b_i \overline{x}_i[K]} \leq ratio^{opt} + B/V \quad , \quad \lim_{K\to\infty} \sum_{i=1}^M c_{il} \overline{x}_i[K] \ \leq \ d_l \ \ \forall l \in \{1, \ldots, L\}$$

for some finite constant B, where $ratio^{opt}$ *is the optimal value of the objective function, and* $\overline{x}_i[K] \triangleq \frac{1}{K} \sum_{k=0}^{K-1} x_i[k]$*. Thus, the limiting time average satisfies the constraints and is within* B/V *from optimality. Your answer should solve for values* $\phi_i[k]$ *such that on frame k we choose* $(x_1[k], \ldots, x_M[k])$ *over* $\mathcal{A}$ *to minimize* $\frac{\phi_0[k] + \sum_{i=1}^M \phi_i[k] x_i[k]}{b_0 + \sum_{i=1}^M b_i x_i[k]}$*. Note: It can be shown a*

solution is: Define $\mathcal{I} \triangleq \{i \in \{1, \ldots, M\} | b_i = 0\}$ *and* $\mathcal{J} \triangleq \{j \in \{1, \ldots, M\} | b_j > 0\}$. *For all* $i \in \mathcal{I}$, *choose* $x_i[k] = 0$ *if* $\phi_i[k] \geq 0$, *and* $x_i[k] = 1$ *if* $\phi_i[k] < 0$. *Next, temporarily select* $x_j[k] = 0$ *for all* $j \in \mathcal{J}$. *Then rank order the indices* $j \in \mathcal{J}$ *from smallest to largest value of* $\phi_j[k]/b_j$, *and, using this order, greedily change* $x_j[k]$ *from* 0 *to* 1 *if it improves the solution.*

b) Note that the case $b_0 = 1$ *and* $b_i = 0$ *for* $i \neq 0$ *is a linear program. Give an explicit decision rule for each* $x_i[k]$ *in this case (the solution should be separable for each* $i \in \{1, \ldots, M\}$*).*

9.3 Reacting to Randomly Observed Events

Consider a problem with general attributes $y_0[k], y_1[k], \ldots, y_L[k]$ and frame size $T[k]$ for each frame k, as in Section 9.2. However, now assume the controller observes a random event $\omega[k]$ at the beginning of each frame k, and this can influence attributes and frame sizes. The value of $\omega[k]$ can represent a vector of channel states and/or prices observed for frame k. Assume $\{\omega[k]\}_{k=0}^{\infty}$ is independent and identically distributed (i.i.d.) over frames. The controller chooses an action $\alpha[k] \in \mathcal{A}(\omega[k])$, where the action space $\mathcal{A}(\omega[k])$ possibly depends on the observed $\omega[k]$. Values of $(T[k], y_0[k], \ldots, y_L[k])$ are conditionally independent of the past given the current $\omega[k]$ and $\alpha[k]$, with mean values:

$$\hat{y}_l(\omega[k], \alpha[k]) = \mathbb{E}\left[y_l[k] | \omega[k], \alpha[k]\right] \quad , \quad \hat{T}(\omega[k], \alpha[k]) = \mathbb{E}\left[T[k] | \omega[k], \alpha[k]\right]$$

The goal is to solve the following optimization problem:

$$\text{Minimize:} \quad \overline{y}_0/\overline{T} \tag{9.48}$$

$$\text{Subject to:} \quad \overline{y}_l/\overline{T} \leq c_l \ \forall l \in \{1, \ldots, L\} \tag{9.49}$$

$$\alpha[k] \in \mathcal{A}(\omega[k]) \ \forall k \in \{0, 1, 2, \ldots\} \tag{9.50}$$

As before, the constraints $\overline{y}_l \leq c_l \overline{T}$ are satisfied via virtual queues $Q_l[k]$ for $l \in \{1, \ldots, L\}$:

$$Q_l[k+1] = \max[Q_l[k] + y_l[k] - c_l T[k], 0] \tag{9.51}$$

The random $\omega[k]$ observations make this problem more complex that those considered in previous sections of this chapter. We present two different algorithms from [25][30].

Algorithm 1:

Every frame k, observe $\boldsymbol{Q}[k]$ and $\omega[k]$ and choose $\alpha[k] \in \mathcal{A}(\omega[k])$ to minimize the following ratio of expectations:

$$\frac{\mathbb{E}\left[V\hat{y}_0(\omega[k],\alpha[k]) + \sum_{l=1}^{L} Q_l[k]\hat{y}_l(\omega[k],\alpha[k])|\boldsymbol{Q}[k]\right]}{\mathbb{E}\left[\hat{T}(\omega[k],\alpha[k])|\boldsymbol{Q}[k]\right]} \tag{9.52}$$

Then update the virtual queues via (9.51).

Algorithm 2:

Define $\theta[0] = 0$, and define $\theta[k]$ for $k \in \{1,2,3,\ldots\}$ as a running ratio of averages over past frames:

$$\theta[k] \triangleq \sum_{i=0}^{k-1} y_0[i] / \sum_{i=0}^{k-1} T[i] \tag{9.53}$$

Then every frame k, observe $\boldsymbol{Q}[k]$, $\theta[k]$, and $\omega[k]$, and choose $\alpha[k] \in \mathcal{A}(\omega[k])$ to minimize the following function:

$$V[\hat{y}_0(\omega[k],\alpha[k]) - \theta[k]\hat{T}(\omega[k],\alpha[k])] + \sum_{l=1}^{L} Q_l[k][\hat{y}_l(\omega[k],\alpha[k]) - c_l\hat{T}(\omega[k],\alpha[k])] \tag{9.54}$$

Then update the virtual queues via (9.51).

Comparison of Algorithms 1 and 2

Both algorithms are introduced and analyzed in [25][30], where they are shown to satisfy the desired constraints and yield an optimality gap of $O(1/V)$. Algorithm 1 can be analyzed in a manner similar to the proof of Theorem 1, and has the same tradeoff with V as given

in that theorem. However, the ratio of expectations (9.52) is *not necessarily* minimized by observing $\omega[k]$ and choosing $\alpha[k] \in \mathcal{A}(\omega[k])$ to minimize the deterministic ratio given $\omega[k]$. In fact, the minimizing policy depends on the (typically unknown) probability distribution for $\omega[k]$. A more complex *bisection algorithm* is needed for implementation, as specified in [25][30].

Algorithm 2 is much simpler and involves a greedy selection of $\alpha[k] \in \mathcal{A}(\omega[k])$ based only on observation of $\omega[k]$, without requiring knowledge of the probability distribution for $\omega[k]$. However, its mathematical analysis does not yield as explicit information regarding convergence time as does Algorithm 1. Further, it requires a running average to be kept starting at frame 0, and hence may not be as adaptive when system statistics change. A more adaptive approximation of Algorithm 2 would define the average $\theta[k]$ over a moving window of some fixed number of frames, or would use an exponentially decaying average.

Both algorithms reduce to the following simplified *drift-plus-penalty* rule in the special case when the frame size $\hat{T}(\omega[k], \alpha[k])$ is a fixed constant T for all $\omega[k], \alpha[k]$: Every frame k, observe $\omega[k]$ and $\boldsymbol{Q}[k]$ and choose $\alpha[k] \in \mathcal{A}(\omega[k])$ to minimize:

$$V\hat{y}_0(\omega[k], \alpha[k]) + \sum_{l=1}^{L} Q_l[k]\hat{y}_l(\omega[k], \alpha[k]) \tag{9.55}$$

Then update the virtual queues via (9.51). This special case algorithm was developed in [29] to treat systems with fixed size time slots.

A simulation comparison of the algorithms is given in [30]. The next subsection describes an application to energy-aware computation and transmission in a wireless smart phone. Exercise 7 considers opportunistic scheduling where wireless transmissions can be deferred by waiting for more desirable channels. Exercise 8 considers an example of price-aware energy consumption for a network server that can process computational tasks or outsource them to another server.

9.3.1 Efficient Computation and Transmission for a Wireless Smart Device

Consider a wireless smart device (such as a smart phone or sensor) that always has tasks to process. Each task involves a computation operation, followed by a transmission operation over a wireless channel. On each frame k, the device takes a new task and looks at its *meta-data* $\beta[k]$, being information that characterizes the task in terms of its computational and transmission requirements. Let d represent the time required to observe this meta-data. The device then chooses a *computational processing mode* $m[k] \in \mathcal{M}(\beta[k])$, where $\mathcal{M}(\beta[k])$ is the set of all mode options under $\beta[k]$. The mode $m[k]$ and meta-data $\beta[k]$ affect a computation time $D_{comp}[k]$, computation energy $e_{comp}[k]$, computation quality $q[k]$, and generate $A[k]$ bits for transmission over the channel. The expectations of $D_{comp}[k]$, $e_{comp}[k]$, $q[k]$ are:

$$\begin{aligned}\hat{D}_{comp}(\beta[k], m[k]) &= \mathbb{E}\left[D_{comp}[k]|\beta[k], m[k]\right] \\ \hat{e}_{comp}(\beta[k], m[k]) &= \mathbb{E}\left[e_{comp}[k]|\beta[k], m[k]\right] \\ \hat{q}(\beta[k], m[k]) &= \mathbb{E}\left[q[k]|\beta[k], m[k]\right]\end{aligned}$$

For example, in a wireless sensor, the mode $m[k]$ may represent a particular sensing task, where different tasks can have different qualities and thus incur different energies, times, and $A[k]$ bits for transmission. The full conditional distribution of $A[k]$, given $\beta[k]$, $m[k]$, will play a role in the transmission stage (rather than just its conditional expectation).

The $A[k]$ units of data must be transmitted over a wireless channel. Let $S[k]$ be the state of the channel on frame k, and assume $S[k]$ is constant for the duration of the frame. We choose a *transmission mode* $g[k] \in \mathcal{G}$, yielding a transmission time $D_{tran}[k]$ and transmission energy $e_{tran}[k]$ with expectations that depend on $S[k]$, $g[k]$, and $A[k]$. Define random event $\omega[k] = (\beta[k], S[k])$ and action $\alpha[k] = (m[k], g[k])$. We can then define expectation functions $\hat{D}_{tran}(\omega[k], \alpha[k])$ and $\hat{e}_{tran}(\omega[k], \alpha[k])$ by:

$$\hat{D}_{tran}(\omega[k], \alpha[k]) = \mathbb{E}\left[D_{tran}[k]|\omega[k], \alpha[k]\right] \quad , \quad \hat{e}_{tran}(\omega[k], \alpha[k]) = \mathbb{E}\left[e_{tran}[k]|\omega[k], \alpha[k]\right]$$

where the above expectations are defined via the *conditional distribution* associated with the number of bits $A[k]$ at the computation output, given the meta-data $\beta[k]$ and computation

mode $m[k]$ selected in the computation phase (where $\beta[k]$ and $m[k]$ are included in the $\omega[k]$, $\alpha[k]$ information). The total frame size is thus $T[k] = d + D_{comp}[k] + D_{tran}[k]$.

The goal is to maximize frame processing quality per unit time $\overline{q}/\overline{T}$ subject to a processing rate constraint of $1/\overline{T} \geq \lambda$, and subject to an average power constraint $(\overline{e}_{comp}+\overline{e}_{tran})/\overline{T} \leq P_{av}$ (where λ and P_{av} are given constants). This fits the general framework with observed random events $\omega[k] = (\beta[k], S[k])$, control actions $\alpha[k] = (m[k], g[k])$, and action space $\mathcal{A}(\omega[k]) = \mathcal{M}(\beta[k]) \times \mathcal{G}$. We can define $y_0[k] = -q[k]$, $y_1[k] = T[k] - 1/\lambda$, and $y_2[k] = e_{comp}[k] + e_{tran}[k] - P_{av}T[k]$, and solve the problem of minimizing $\overline{y}_0/\overline{T}$ subject to $\overline{y}_1 \leq 0$ and $\overline{y}_2 \leq 0$. To do so, let $Q[k]$ and $Z[k]$ be virtual queues for the two constraints:

$$Q[k+1] = \max[Q[k] + T[k] - 1/\lambda, 0] \tag{9.56}$$

$$Z[k+1] = \max[Z[k] + e_{comp}[k] + e_{tran}[k] - P_{av}T[k], 0] \tag{9.57}$$

Using Algorithm 2, we define $\theta[0] = 0$ and $\theta[k]$ for $k \in \{1, 2, \ldots\}$ by (9.53). The Algorithm 2 minimization (9.54) amounts to observing $(\beta[k], S[k])$, $Q[k]$, $Z[k]$, and $\theta[k]$ on each frame k and choosing $m[k] \in \mathcal{M}(\beta[k])$ and $g[k] \in \mathcal{G}$ to minimize:

$$\begin{aligned} &V[-\hat{q}(\beta[k], m[k]) - \theta[k](d + \hat{D}_{comp}(\beta[k], m[k]) + \hat{D}_{tran}(\omega[k], \alpha[k]))] \\ &+Q[k][d + \hat{D}_{comp}(\beta[k], m[k]) + \hat{D}_{tran}(S[k], g[k]) - 1/\lambda] \\ &+Z[k][\hat{e}_{comp}(\beta[k], m[k]) + \hat{e}_{tran}(\omega[k], \alpha[k]) - P_{av}(d + \hat{D}_{comp}(\beta[k], m[k]) + \hat{D}_{tran}(\omega[k], \alpha[k]))] \end{aligned}$$

The computation and transmission operations are coupled and cannot be separated. This yields the following algorithm: Every frame k:

- Observe $\omega[k] = (\beta[k], S[k])$ and values $Q[k]$, $Z[k]$, $\theta[k]$. Then *jointly* choose action $m[k] \in \mathcal{M}(\beta[k])$ and $g[k] \in \mathcal{G}$, for a combined action $\alpha[k] = (m[k], g[k])$, to minimize:

$$\begin{aligned} &-V\hat{q}(\beta[k], m[k]) + \hat{D}_{comp}(\beta[k], m[k])[Q[k] - V\theta[k] - P_{av}Z[k]] + Z[k]\hat{e}_{comp}(\beta[k], m[k]) \\ &+\hat{D}_{tran}(\omega[k], \alpha[k])[Q[k] - V\theta[k] - P_{av}Z[k]] + Z[k]\hat{e}_{tran}(\omega[k], \alpha[k]) \end{aligned}$$

- (Updates) Update $Q[k]$, $Z[k]$, $\theta[k]$ via (9.56), (9.57), and (9.53).

Exercise 9 shows the algorithm can be implemented without $\theta[k]$ if the goal is changed to maximize $\overline{q}$, rather than $\overline{q}/\overline{T}$. Further, the computation and transmission decisions can be *separated* if the system is modified so that the bits generated from computation are handed to a separate transmission layer for eventual transmission over the channel, rather than requiring transmission on the same frame, similar to the structure used in Exercise 8.

9.3.2 Exercises for Section 9.3

Exercise 7. *(Energy-Efficient Opportunistic Scheduling [29]) Consider a wireless device that operates over fixed size time slots $k \in \{0, 1, 2, \ldots\}$. Every slot k, new data of size $A[k]$ bits arrives and is added to a queue. The data must eventually be transmitted over a time-varying channel. At the beginning of every slot k, a controller observes the* channel state *$S[k]$ and allocates power $p[k]$ for transmission, enabling transmission of $\mu[k]$ bits, where $\mu[k] = \hat{\mu}(S[k], p[k])$ for some given function $\hat{\mu}(S, p)$. Assume $p[k]$ is chosen so that $0 \leq p[k] \leq P_{max}$ for some constant $P_{max} > 0$. We want to minimize average power $\overline{p}$ subject to supporting all data, so that $\overline{A} \leq \overline{\mu}$. Treat this as a problem with all frames equal to 1 slot, observed random events $\omega[k] = (A[k], S[k])$, and actions $\alpha[k] = p[k] \in [0, P_{max}]$. Design an appropriate queue update and power allocation algorithm, using the policy structure* (9.55).

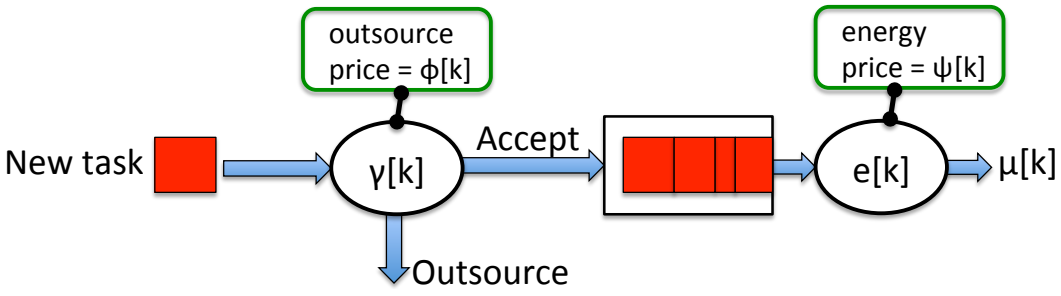

Figure 9.10: The client/server system of Exercise 8, with decision variables $(\gamma[k], e[k])$ and observed prices $(\phi[k], \psi[k])$.

Exercise 8. *(Energy Prices and Network Outsourcing) Consider a computer server that operates over fixed length time slots $k \in \{0, 1, 2, \ldots\}$. Every slot k, a new task arrives and has size $S[k]$ (if no task arrives then $S[k] = 0$). The server decides to either accept the task, or outsource it to another server (see Fig. 9.10). Let $\gamma[k]$ be a binary decision variable that*

is 1 if the server accepts the task on slot k, and zero else. Define $A[k] = \gamma[k]S[k]$ as the total workload admitted on slot k, which is added to the queue of work to be done. Let $\phi[k]$ be the (possibly time-varying) cost per unit size for outsourcing, so the outsourcing cost is $c_{out}[k] = \phi[k](1-\gamma[k])S[k]$. Every slot k, the server additionally decides to process some of its backlog by purchasing an amount of energy $e[k]$ at a per unit energy price $\psi[k]$, serving $\mu[k] = \hat{\mu}(e[k])$ units of backlog with cost $c_{energy}[k] = \psi[k]e[k]$, where $\hat{\mu}(e)$ is some given function. Assume $e[k]$ is chosen in some interval $0 \leq e[k] \leq e_{max}$. The goal is to minimize time average cost $\overline{c}_{out} + \overline{c}_{energy}$ subject to supporting all tasks, so that $\overline{A} \leq \overline{\mu}$. Treat this as a problem with all frames equal to 1 slot, observed random events $\omega[k] = (S[k], \phi[k], \psi[k])$, actions $\alpha[k] = (\gamma[k], e[k])$, and action space $\mathcal{A}(\omega[k])$ being the set of all (γ, e) such that $\gamma \in \{0, 1\}$ and $0 \leq e \leq e_{max}$. Design an appropriate queue and state the dynamic algorithm (9.55) *for this problem. Show that it can be implemented without knowledge of the probability distribution for $(S[k], \phi[k], \psi[k])$, and that the $\gamma[k]$ and $e[k]$ decisions are separable.*

Exercise 9. *Consider the same problem of Section 9.3.1, with the modified objective of maximizing $\overline{q}$ subject to $1/\overline{T} \geq \lambda$ and $(\overline{e}_{comp} + \overline{e}_{tran})/\overline{T} \leq P_{av}$. Use the observation from Exercise 4 that this can be viewed as a problem with an "effective" fixed frame size equal to 1, and give an algorithm from the policy structure* (9.55)*.*

9.4 Conclusions

This chapter presents a methodology for optimizing time averages in systems with variable length frames. Applications include energy and quality aware task scheduling in smart phones, cost effective energy management at computer servers, and more. The resulting algorithms are dynamic and often do not require knowledge of the probabilities that affect system events. While the performance theorem of this chapter was stated under simple i.i.d. assumptions, the same algorithms are often provably robust to non-i.i.d. situations, including situations where the events are non-ergodic [25]. Simulations in Section 9.2.4 show examples of how such algorithms adapt to changes in the event probabilities. Exercises in this chapter were included to help readers learn to design dynamic algorithms for their own optimization problems.

The solution technique of this chapter uses the theory of optimization for renewal systems from [25][30], which applies to general problems. Performance for individual problems can often be improved using enhanced techniques, such as using place-holder backlog, exponential Lyapunov functions, LIFO scheduling, and ϵ-persistent service queues, as discussed in [25] and references therein. However, the drift-plus-penalty methodology described in this chapter provides much of the insight needed for these more advanced techniques. It is also simple to work with and typically yields desirable solution quality and convergence time.

Appendix A — Bounded Moment Convergence Theorem

This appendix provides a *bounded moment convergence theorem* that is often more convenient than the standard Lebesgue dominated convergence theorem (see, for example, [42] for the standard Lebesgue dominated convergence theorem). We are unaware of a statement and proof in the literature, so we give one here for completeness. Let $X(t)$ be a random process defined either over non-negative real numbers $t \geq 0$, or discrete time $t \in \{0, 1, 2, \ldots\}$. Recall that $X(t)$ converges *in probability* to a constant x if for all $\epsilon > 0$ we have:

$$\lim_{t\to\infty} Pr[|X(t) - x| > \epsilon] = 0$$

Theorem 2. *Suppose there is a real number x such that $X(t)$ converges to x in probability. Further suppose there are finite constants $C > 0$, $\delta > 0$ such that for all t we have $\mathbb{E}\left[|X(t)|^{1+\delta}\right] \leq C$. Then $\lim_{t\to\infty} \mathbb{E}[X(t)] = x$.*

Proof. Without loss of generality, assume $x = 0$ (else, we can define $Y(t) = X(t) - x$). Fix $\epsilon > 0$. By definition of $X(t)$ converging to 0 in probability, we have:

$$\lim_{t\to\infty} Pr[|X(t)| > \epsilon] = 0 \tag{9.58}$$

Further, for all t we have $\mathbb{E}[X(t)] \leq \mathbb{E}[|X(t)|]$, and so:

$$\mathbb{E}[X(t)] \leq \epsilon + \mathbb{E}[|X(t)| \,|\, |X(t)| > \epsilon]\, Pr[|X(t)| > \epsilon] \tag{9.59}$$

We want to show the final term in the right-hand-side above converges to 0 when $t \to \infty$. To this end, note that for all t we have:

$$\begin{aligned} C &\geq \mathbb{E}\left[|X(t)|^{1+\delta}\right] \\ &\geq \mathbb{E}\left[|X(t)|^{1+\delta} \,|\, |X(t)| > \epsilon\right] Pr[|X(t)| > \epsilon] \\ &\geq \mathbb{E}\left[|X(t)| \,|\, |X(t)| > \epsilon\right]^{1+\delta} Pr[|X(t)| > \epsilon] \qquad (9.60) \end{aligned}$$

where (9.60) follows by Jensen's inequality applied to the conditional expectation of the function $f(|X(t)|)$, where $f(x) = x^{1+\delta}$ is convex over $x \geq 0$. Multiplying inequality (9.60) by $Pr[|X(t)| > \epsilon]^{\delta}$ yields:

$$CPr[|X(t)| > \epsilon]^{\delta} \geq (\mathbb{E}\left[|X(t)| \,|\, |X(t)| > \epsilon\right] Pr[|X(t)| > \epsilon])^{1+\delta} \geq 0$$

Taking a limit of the above as $t \to \infty$ and using (9.58) yields:

$$0 \geq \lim_{t\to\infty} (\mathbb{E}\left[|X(t)| \,|\, |X(t)| > \epsilon\right] Pr[|X(t)| > \epsilon])^{1+\delta} \geq 0$$

It follows that:

$$\lim_{t\to\infty} \mathbb{E}\left[|X(t)| \,|\, |X(t)| > \epsilon\right] Pr[|X(t)| > \epsilon] = 0$$

Using this equality and taking a lim sup of (9.59) yields $\limsup_{t\to\infty} \mathbb{E}\left[X(t)\right] \leq \epsilon$. This holds for all $\epsilon > 0$, and so $\limsup_{t\to\infty} \mathbb{E}\left[X(t)\right] \leq 0$. Similarly, it can be shown that $\liminf_{t\to\infty} \mathbb{E}\left[X(t)\right] \geq 0$. Thus, $\lim_{t\to\infty} \mathbb{E}\left[X(t)\right] = 0$. □

Recall that convergence with probability 1 is stronger than convergence in probability, and so the above result also holds if $\lim_{t\to\infty} X(t) = x$ with probability 1. Theorem 2 can be applied to the case when $\lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} e[k] = \overline{e}$ with probability 1, for some finite constant $\overline{e}$, and when there is a constant C such that $\mathbb{E}\left[e[k]^2\right] \leq C$ for all k. Indeed, one can define $X(K) = \frac{1}{K}\sum_{k=0}^{K-1} e[k]$ for $K \in \{1, 2, 3, \ldots\}$ and use the Cauchy-Schwartz inequality to show $\mathbb{E}\left[X(K)^2\right] \leq C$ for all $K \in \{1, 2, 3, \ldots\}$, and so $\lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[e[k]\right] = \overline{e}$.

References

[1] R. Agrawal and V. Subramanian. Optimality of certain channel aware scheduling policies. *Proc. 40th Annual Allerton Conf. on Communication, Control, and Computing, Monticello, IL*, Oct. 2002.

[2] M. Annavaram, E. Grochowski, and J. Shen. Mitigating amdahl's law through epi throttling. *Proc. 32nd International Symposium on Computer Architecture (ISCA)*, pp. 298-309, June 2005.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] A. Eryilmaz and R. Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Nonlinear Optimization of Communication Systems*, vol. 14, pp. 1514-1524, Aug. 2006.

[5] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *Proc. IEEE INFOCOM*, March 2005.

[6] R. Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, Boston, 1996.

[7] M. Gatzianas, L. Georgiadis, and L. Tassiulas. Control of wireless networks with rechargeable batteries. *IEEE Transactions on Wireless Communications*, vol. 9, no. 2, pp. 581-593, Feb. 2010.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1-149, 2006.

[9] E. Grochowski, R. Ronen, J. Shen, and H. Wang. Best of both latency and throughput. *Proc. IEEE Conf. on Computer Design (ICCD)*, pp. 236-243, Oct. 2004.

[10] I. Hou, V. Borkar, and P. R. Kumar. A theory of QoS for wireless. *Proc. IEEE INFOCOM*, April 2009.

[11] I. Hou and P. R. Kumar. Utility maximization for delay constrained qos in wireless. *Proc. IEEE INFOCOM*, March 2010.

[12] L. Huang and M. J. Neely. Utility optimal scheduling in processing networks. *Proc. IFIP, Performance*, 2011.

[13] L. Huang and M. J. Neely. Utility optimal scheduling in energy harvesting networks. *Proc. Mobihoc*, May 2011.

[14] L. Jiang and J. Walrand. *Scheduling and Congestion Control for Wireless and Processing Networks*. Morgan & Claypool, 2010.

[15] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol. 8, no. 1 pp. 33-37, Jan.-Feb. 1997.

[16] F.P. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness, and stability. *Journ. of the Operational Res. Society*, vol. 49, no. 3, pp. 237-252, March 1998.

[17] H. Kushner and P. Whiting. Asymptotic properties of proportional-fair sharing algorithms. *Proc. 40th Annual Allerton Conf. on Communication, Control, and Computing, Monticello, IL*, Oct. 2002.

[18] C. Li and M. J. Neely. Network utility maximization over partially observable Markovian channels. *Arxiv Technical Report: arXiv:1008.3421v1*, Aug. 2010.

[19] C. Li and M. J. Neely. Network utility maximization over partially observable Markovian channels. *Proc. Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2011.

[20] Q. Li and R. Negi. Scheduling in wireless networks under uncertainties: A greedy primal-dual approach. *Arxiv Technical Report: arXiv:1001:2050v2*, June 2010.

[21] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. *Proc. of 43rd IEEE Conf. on Decision and Control, Paradise Island, Bahamas*, Dec. 2004.

[22] S. H. Low. A duality model of TCP and queue management algorithms. *IEEE Trans. on Networking*, vol. 11, no. 4, pp. 525-536, August 2003.
[23] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, Oct. 2000.
[24] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, LIDS, 2003.
[25] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.
[26] M. J. Neely. Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks. *Proc. IEEE INFOCOM*, 2011.
[27] M. J. Neely. Stability and probability 1 convergence for queueing networks via Lyapunov optimization. *Journal of Applied Mathematics*, vol. 2012, doi:10.1155/2012/831909, 2012.
[28] M. J. Neely. Stock market trading via stochastic network optimization. *Proc. IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010.
[29] M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915-2934, July 2006.
[30] M. J. Neely. Dynamic optimization and learning for renewal systems. *Proc. Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2010.
[31] M. J. Neely and L. Huang. Dynamic product assembly and inventory control for maximum profit. *Proc. IEEE Conf. on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010.
[32] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proc. IEEE INFOCOM*, pp. 1723-1734, March 2005.
[33] M. J. Neely, A. S. Tehrani, and A. G. Dimakis. Efficient algorithms for renewable energy allocation to delay tolerant consumers. *1st IEEE International Conference on Smart Grid Communications*, Oct. 2010.
[34] A. Ribeiro and G. B. Giannakis. Separation principles in wireless networking. *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4488-4505, Sept. 2010.
[35] S. Ross. *Introduction to Probability Models*. Academic Press, 8th edition, Dec. 2002.
[36] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, vol. 50, no. 4, pp. 401-457, 2005.
[37] A. Tang, J. Wang, and S. Low. Is fair allocation always inefficient. *Proc. IEEE INFOCOM*, March 2004.
[38] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transacations on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.
[39] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466-478, March 1993.
[40] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam. Optimal power cost management using stored energy in data centers. *Proc. SIGMETRICS*, June 2011.
[41] W.-H. Wang, M. Palaniswami, and S. H. Low. Application-oriented flow control: Fundamentals, algorithms, and fairness. *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, Dec. 2006.
[42] D. Williams. *Probability with Martingales*. Cambridge Mathematical Textbooks, Cambridge University Press, 1991.
[43] L. Xiao, M. Johansson, and S. P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, vol. 52, no. 7, pp. 1136-1144, July 2004.