Achieving Utility-Delay-Reliability Tradeoff in Stochastic Network Optimization with Finite Buffers

Sucha Supittayapornpong, Michael J. Neely
Department of Electrical Engineering
University of Southern California
Email: supittay@usc.edu, mjneely@usc.edu

***Abstract*—One practical open problem is the development of a distributed algorithm that achieves near-optimal utility using only a finite (and small) buffer size for queues in a stochastic network. This paper studies utility maximization (or cost minimization) in a finite-buffer regime and considers the corresponding delay and reliability (or rate of packet drops) tradeoff. A *floating-queue* algorithm allows the stochastic network optimization framework to be implemented with finite buffers at the cost of packet drops. Further, the buffer size requirement is significantly smaller than previous works in this area. With a finite buffer size of B packets, the proposed algorithm achieves within $O(e^{-B})$ of the optimal utility while maintaining average per-hop delay of $O(B)$ and an average per-hop drop rate of $O(e^{-B})$ in steady state. From an implementation perspective, the floating-queue algorithm requires little modification of the well-known Drift-Plus-Penalty policy (including MaxWeight and Backpressure policies). As a result, the floating-queue algorithm inherits the distributed and low complexity nature of these policies.**

I. Introduction

Stochastic network optimization is a general framework for solving a network optimization problem with randomness [1]. The framework generates a control algorithm that achieves a specified objective, such as minimizing power cost or maximizing throughput utility. It is assumed that the network has random states that evolve over discrete time. Every time slot, a network controller observes the current network state and makes a control decision. The network state and control decision together incur some cost and, at the same time, serves some amount of traffic from network queues. The algorithm is designed to greedily minimize a *drift-plus-penalty* expression every slot. This greedy procedure is known to minimize time average network cost subject to queue stability.

This general framework has been used to solve several network optimization problems such as network routing [2], throughput maximization [3], dynamic power allocation [4], quality of information maximization [5]. The framework yields low-complexity algorithms which do not require any statistical knowledge of the network states. Therefore, these algorithms are easy to implement and are robust to environment changes. Further, they achieve an $[O(1/V), O(V)]$ utility-delay tradeoff, where $V > 0$ is a parameter that can be chosen as desired to achieve a specific operating point on the $[O(1/V), O(V)]$ utility-delay tradeoff curve.

Prior works attempt to improve network delay without sacrificing reliability, where reliability is measured by the rate of packet drops. Previous works [6] and [7] use an exponential Lyapunov function and assumed knowledge of an ϵ parameter, where ϵ measures a distance associated with the optimal operation point. They achieve an optimal $[O(1/V), O(\log(V))]$ utility-delay tradeoff. A simpler methodology allows packet drops in order to obtain an $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff [8], [9]. In [8], a steady state behavior is observed to learn a placeholder parameter to achieve the tradeoff in steady state. However, the algorithm does not gracefully adapt to changes of the network state distribution. It would need another mechanism to sense changes and then recompute a new placeholder parameter with each change. The Last-In-First-Out (LIFO) queue discipline is employed to resolve this issue [9]. However, these works, which achieve average queue size that grows logarithmically in V, still assume the availability of infinite buffer space [7]–[9].

A practical implementation of the LIFO scheme is developed in [10]. The work in [10] also introduces a *floating-queue* algorithm, operating under the LIFO scheme, to deal with finite buffers. The algorithm in [10] is heuristic, and it is not clear how to analyze its behavior. The current work is inspired by this floating queue idea of [10] and adopts the same "floating queue" terminology, even though the floating-queue algorithm developed here is different from [10]. Indeed, the floating queue technique of this paper operates under the First-In-First-Out (FIFO) scheme. It splits each queue into two queues (one for *real* and one for *fake* packets) and yields analytical guarantees on utility, delay, and packet drops.

Several backpressure approaches [11]–[13] attempt to improve network delay. However, those focus on specific aspects and do not have the theoretical utility-delay tradeoff. Stochastic network optimization with finite buffers has been studied previously in [14]. That work uses a non-standard Lyapunov function and knowledge of an ϵ parameter to derive an upper bound on the required buffer size. However, the ϵ parameter can be difficult to determine in practice, and the resulting utility-delay tradeoff is still $[O(1/V), O(V)]$. An implementation of that work is studied in [15].

This paper develops a floating-queue approach to general stochastic network optimization with finite buffers. Our algorithm is inspired by finite buffer heuristics in [10] and the steady state analysis in [8]. We propose the floating-queue algorithm to solve the learning issue in [8]. The result obtains the best of both worlds: It achieves the desired steady state performance but is just as adaptive to network changes as

This work is supported by the NSF under Career grant CCF-0747525.

LIFO scheduling. For finite buffers of size B, deviation from utility optimality is shown to decrease like $O(e^{-B})$ and packet drops are shown to have rate $O(e^{-B})$, while average per-hop delay is $O(B)$.

This paper is organized as follows. The system model is described in Section II. Section III describes the standard drift-plus-penalty approach. The floating queue algorithm is introduced in Section IV. Performance of the floating-queue algorithm is analyzed in Section V and is validated by simulation in Section VI. Section VII concludes the paper.

II. System Model

The network model of this paper is similar to that of [8]. Consider a network with N queues that evolve in discrete (slotted) time $t \in \{0, 1, 2, \dots\}$. At each time slot, a *network controller* observes the current *network state* before making a *decision*. The goal of the controller is to minimize a time average cost subject to network stability. An example of time average cost is average power incurred over the network. Utility maximization can be treated by defining the slot-t cost as -1 times a slot-t reward. An example utility maximization problem is to maximize time average network throughput. The rest of the paper deals with cost minimization, with the understanding that this can also treat utility maximization.

A. Network State

The network experiences randomness every time slot. This randomness is called the *network state* and can represent a vector of channel conditions and/or random arrivals for slot t. Assume there are M different network states. Define $\mathcal{S} = \{s_1, s_2, \dots, s_M\}$ as the set of all possible states. Let $S(t)$ denote the network state experienced by the network at time t. Let $\pi_m \in [0, 1]$ be the steady state probability that $S(t) = s_m$, i.e., $\pi_m = \mathbb{P}\{S(t) = s_m\}$. For simplicity, it is assumed that $S(t)$ is independent and identically distributed (i.i.d.) over slots. The same results can be shown in the general case of ergodic but non-i.i.d. processes (see [8]). The network controller can observe $S(t)$ before making the slot-t decision, but the π_m probabilities are not known to the controller.

B. Control Decision

Every time slot, the network controller chooses a decision from a set of feasible actions which depends on the current network state. Formally, define $\mathcal{X}^{S(t)}$ as the decision set depending on $S(t)$, and let $x(t)$ denote the decision chosen by the controller at time t, where $x(t) \in \mathcal{X}^{S(t)}$. Assume the action set $\mathcal{X}^{s_m}$ is finite for every $s_m \in \mathcal{S}$. On slot t, these control decision and network state $(x(t), S(t))$ affects the network in 2 aspects:

1) A cost is incurred. The cost is $f(t) \triangleq f(x(t), S(t)) : \mathcal{X}^{S(t)} \to \mathbb{R}$. An example cost is energy expenditure. Another example is -1 times the amount of newly admitted packets.

2) Queues are served. The service variables are $\mu_{ij}(t)$, representing the integer amount of packets taken from queue i and transmitted to queue j, for all $i, j \in \mathcal{N} \triangleq \{1, \dots, N\}$. This is determined by a function $\mu_{ij}(t) \triangleq \mu_{ij}(x(t), S(t)) : \mathcal{X}^{S(t)} \to \mathbb{Z}_+$. Further, the decision admits $\mu_{0i}(t) \triangleq \mu_{0i}(x(t), S(t)) : \mathcal{X}^{S(t)} \to \mathbb{Z}_+$ integer amount of exogenous packets to queue $i \in \mathcal{N}$. Packets depart from the network at queue $j \in \mathcal{N}$ with an integer amount $\mu_{j0}(t) \triangleq \mu_{j0}(x(t), S(t)) : \mathcal{X}^{S(t)} \to \mathbb{Z}_+$. Note that we set $\mu_{ii}(t) = 0$ for all $i \in \mathcal{N} \cup \{0\}$ and for all t. This is illustrated in Figure 1.

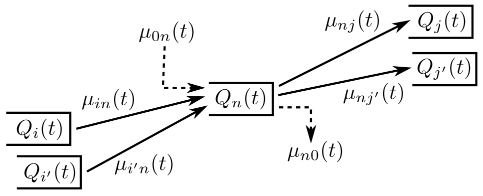

Fig. 1. Arrivals and services at a standard queue

For every $s_m \in \mathcal{S}$, we assume functions $f(\cdot, s_m)$ and $\mu_{ij}(\cdot, s_m)$ for $i, j \in \mathcal{N} \cup \{0\}$ are time-invariant, and magnitudes of $\sum_{i=0}^{N} \mu_{in}(\cdot, s_m)$ and $\sum_{j=0}^{N} \mu_{nj}(\cdot, s_m)$ are upper bounded by constant $\delta^{(\max)} \in (0, \infty)$ for every $n \in \mathcal{N}$. Furthermore, the network optimization is assumed to satisfy the following *Slater condition* [8]: Let $|\mathcal{X}|$ be the cardinality of set $\mathcal{X}$. For every $s_m \in \mathcal{S}$ and $k \in \{1, \dots, |\mathcal{X}^{s_m}|\}$, there exist probabilities $\zeta_k^{s_m}$ (such that $\sum_{k=1}^{|\mathcal{X}^{s_m}|} \zeta_k^{s_m} = 1$ for all $s_m \in \mathcal{S}$) that define a *stationary and randomized algorithm*. Whenever the network controller observes $S(t) = s_m$, the stationary and randomized algorithm chooses action $x_k^{s_m}$ with conditional probability $\zeta_k^{s_m}$. The Slater condition assumes there exists such a stationary and randomized algorithm satisfying:

$$\sum_{m=1}^{M} \sum_{k=1}^{|\mathcal{X}^{s_m}|} \pi_m \zeta_k^{s_m} \left[\sum_{i=0}^{N} \mu_{in}(x_k^{s_m}, s_m) - \sum_{j=0}^{N} \mu_{nj}(x_k^{s_m}, s_m) \right] \le -\eta \qquad \text{for all } n \in \mathcal{N},$$

for some $\eta > 0$. In fact, this assumption is the standard Stater condition of convex optimization [16].

C. Standard Queue

The network consists of N *standard queues*. Let $Q_n(t)$ denote the backlog in queue n at time t, and let $Q(t) \triangleq (Q_1(t), \dots, Q_N(t))$ be the vector of these backlogs. The backlog dynamic of queue $n \in \{1, \dots, N\}$ is

$$Q_n(t+1) = \max\left[Q_n(t) - \sum_{j=0}^{N} \mu_{nj}(t), 0\right] + \sum_{i=0}^{N} \mu_{in}(t). \quad (1)$$

When there are not enough packets in a queue, i.e, $Q_n(t) < \sum_{j=0}^{N} \mu_{nj}(t)$, blank packets are used to fill up transmissions.

D. Stochastic Formulation

The controller seeks to minimize the expected time-average cost while maintaing queue stability. The expected time average cost is defined by

$$\bar{f} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[f(t)],$$

```
Initialization: $Q(0) = \mathbf{0}$
for $t \in \{0, 1, 2, \dots\}$ do
    Observer $S(t)$ and $Q(t)$.
    Choose $x(t)$ that solves (3).
    Update $Q_n(t+1)$ according to (1) $\forall n \in \mathcal{N}$.
end for
```

Fig. 2. The drift-plus-penalty algorithm

and the queue stability is satisfied when

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n=1}^{N} \mathbb{E}\left[Q_n(t)\right] < \infty.$$

The *stochastic network optimization problem* is

$$\begin{aligned} \text{Minimize} \quad & \bar{f} \\ \text{Subject to} \quad & \text{queue stability.} \end{aligned} \tag{2}$$

III. Drift-Plus-Penalty Method

A. Drift-Plus-Penalty Method

The drift-plus-penalty method of [1] can solve problem (2) via a greedy decision at each time slot that does not require knowledge of the steady state probabilities. The method has parameter $V \geq 0$. In the special case of $V = 0$, this policy is also called "MaxWeight" or "Backpressure":

Drift-Plus-Penalty Policy: At every time $t \in \{0, 1, 2, \dots\}$, the network controller observes network state $S(t)$ and backlog vector $Q(t)$. Decision $x(t) \in \mathcal{X}^{S(t)}$ is chosen to solve:

$$\begin{aligned} \text{Minimize} \quad & Vf(x(t), S(t)) \\ & + \sum_{n=1}^{N} Q_n(t) \left[\sum_{i=0}^{N} \mu_{in}(x(t), S(t)) - \sum_{j=0}^{N} \mu_{nj}(x(t), S(t)) \right] \\ \text{Subject to} \quad & x(t) \in \mathcal{X}^{S(t)}. \end{aligned} \tag{3}$$

Depending on the separability structure of problem (3), it can be decomposed to smaller subproblems that can be solved distributively. The algorithm is summarized in Figure 2.

It has been shown in [1] that

$$f^{(\text{DPP})} \leq f^{(\text{opt})} + O(1/V) \tag{4}$$

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n=1}^{N} \mathbb{E}\left[Q_n(t)\right] = O(V), \tag{5}$$

where $f^{(\text{DPP})}$ is the expected time average cost achieved by the drift-plus-penalty policy, and $f^{(\text{opt})}$ is the optimal cost of problem (2). The inequality (4) implies that the drift-plus-penalty policy achieves cost within $O(1/V)$ of the optimal cost, which can be made as small as desired by choosing a sufficiently large value of V. The equality (5) implies that average queue backlog grows linearly with V. Applying Little's law gives the $[O(1/V), O(V)]$ utility-delay tradeoff (see [17] for a standard description of Little's law).

Notice that the drift-plus-penalty algorithm assumes infinite buffer size at each queue, even though the average queue size is bounded by $O(V)$.

B. Deterministic Problem

In order to consider a finite buffer regime, the steady-state behavior of the drift-plus-penalty algorithm is considered. In [8], the stochastic problem (2) is shown to have an associated deterministic problem as follows:

$$\text{Minimize} \quad V \sum_{m=1}^{M} \pi_m f(x^{s_m}, s_m) \tag{6}$$

$$\begin{aligned} \text{Subject to} \quad & \sum_{m=1}^{M} \pi_m \sum_{i=0}^{N} \mu_{in}(x^{s_m}, s_m) \\ & \leq \sum_{m=1}^{M} \pi_m \sum_{j=0}^{N} \mu_{nj}(x^{s_m}, s_m) \quad \forall n \in \mathcal{N} \qquad (7) \\ & x^{s_m} \in \mathcal{X}^{s_m} \quad \forall m \in \{1, \dots, M\}. \end{aligned}$$

Let $\gamma = (\gamma_1, \dots, \gamma_N)$ be a vector of dual variables associated with constraint (7). The dual function of problem (6) is defined as:

$$\begin{aligned} g(\gamma) = \sum_{m=1}^{M} \pi_m \inf_{x^{s_m} \in \mathcal{X}^{s_m}} \Bigg\{ & Vf(x^{s_m}, s_m) \\ + \sum_{n=1}^{N} \gamma_n & \left[\sum_{i=0}^{N} \mu_{in}(x^{s_m}, s_m) - \sum_{j=0}^{N} \mu_{nj}(x^{s_m}, s_m) \right] \Bigg\}. \end{aligned} \tag{8}$$

This dual function (8) is concave. Therefore, the following dual problem is a convex optimization problem:

$$\begin{aligned} \text{Maximize} \quad & g(\gamma) \\ \text{Subject to} \quad & \gamma \in \mathbb{R}_+^N. \end{aligned} \tag{9}$$

Let $\gamma^{V*} = (\gamma_1^{V*}, \dots, \gamma_N^{V*})$ be a vector of Lagrange multipliers, which solves the dual problem (9) with parameter V. The following theorem from [9] describes a steady state property of the drift-plus-penalty algorithm:

Theorem 1: Suppose γ^{V*} is unique, the Slater condition holds, and the dual function $g(\gamma)$ satisfies:

$$g(\gamma^{V*}) \geq g(\gamma) + L\left\|\gamma^{V*} - \gamma\right\| \quad \text{for all } \gamma \in \mathbb{R}_+^N,$$

for some constant $L > 0$, independent of V. Then under the drift-plus-penalty policy, there exist constants D, K, c^*, independent of V, such that for any $\beta \geq 0$, the following upper bound holds

$$\mathcal{P}(D, K\beta) \leq c^* e^{-\beta}, \tag{10}$$

where

$$\begin{aligned} & \mathcal{P}(D, K\beta) \\ & \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{P}\left\{\exists n, \left|Q_n(t) - \gamma_n^{V*}\right| > D + K\beta\right\}. \end{aligned} \tag{11}$$

Proof: Please see the full proof in [8]. ■

As all transmissions, admissions, and departures are integers, the queue vector has a countably infinite number of possibilities, and under a mild ergodic assumption the steady state distribution of $\{Q(t) : t \geq 0\}$ exists. In this ergodic case, the $\mathcal{P}(D, K\beta)$ value in (11) becomes the steady state probability that backlog deviates more than $D + K\beta$ away from

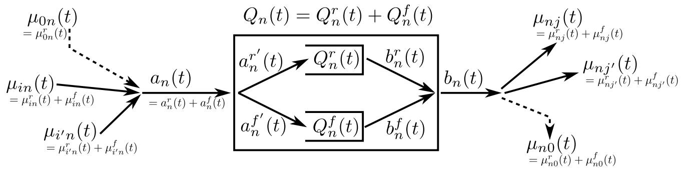

Fig. 3. Transformation of a standard queue to a floating queue

the vector of Lagrange multipliers. Note that the probability in (10) vanishes exponentially in β. This implies that, in steady state, a large portion of arrivals and services occurs when the queue backlog vector is close to the Lagrange multiplier vector γ^{V*}. Thus, if we can admit and serve this portion of traffic using finite-buffer queues, the network still operates near its optimal point.

IV. Floating-Queue Algorithm

In this section, the floating-queue algorithm is presented as a way to implement the drift-plus-penalty algorithm using finite buffers. The algorithm preserves the dynamics of the drift-plus-penalty algorithm and hence inherits several of its performance guarantees.

Recall that standard queue $n \in \mathcal{N}$ has dynamic (1). To simplify notation, let $a_n(t) \triangleq \sum_{i=0}^N \mu_{in}(t)$ denote aggregated arrivals to queue n, and let $b_n(t) \triangleq \sum_{j=0}^N \mu_{nj}(t)$ denote aggregated services from queue n at time t. This implies that $\delta^{(\max)}$ upper bounds both $a_n(t)$ and $b_n(t)$. The dynamic (1) can be written as

$$Q_n(t+1) = \max\left[Q_n(t) - b_n(t), 0\right] + a_n(t).$$

For the rest of this paper, the above dynamic is considered for a standard queue. Note that $a_n(t)$ and $b_n(t)$ are fully determined after knowing all $\mu_{ij}(t)$ from the drift-plus-penalty algorithm.

A. Queue Transformation

In the floating-queue algorithm, each standard queue $n \in \mathcal{N}$ of Section II-C is a combination of a *real queue* and a *fake queue*. The real queue has buffer size B for storing *real packets*. The fake queue contains *fake packets* and only requires a counter to implement. Let $Q^r_n(t)$ and $Q^f_n(t)$ denote respectively the amount of backlogs in the real queue and the fake queue of the standard queue n. Define $Q^r(t) \triangleq (Q^r_1(t), \dots, Q^r_N(t))$ and $Q^f(t) \triangleq (Q^f_1(t), \dots, Q^f_N(t))$ as vectors of real and fake queue backlogs. We use the term *floating queue* n to refer to the 2-queue combination consisting of real and fake queues n.

B. Real and Fake Parts of Arrivals and Services

At each queue $n \in \mathcal{N}$, let $b^r_n(t)$ and $b^f_n(t)$ denote the aggregated real and fake serviced packets at time t. the floating-queue algorithm always serves real packets before fake packets as:

$$b^r_n(t) = \min\left[Q^r_n(t), b_n(t)\right] \tag{12}$$

$$b^f_n(t) = b_n(t) - b^r_n(t). \tag{13}$$

It is easy to see that $b_n(t) = b^r_n(t) + b^f_n(t)$. Also, $b^r_n(t)$ and $b^f_n(t)$ are fully determined, since $b_n(t)$ and $Q^r_n(t)$ are known.

To differentiate the real and fake packets in the drift-plus-penalty variable $\mu_{ij}(t)$, let $\mu^r_{ij}(t)$ and $\mu^f_{ij}(t)$ denote the real and fake parts of $\mu_{ij}(t)$ for $i,j \in \mathcal{N} \cup \{0\}$. All $\mu_{ij}(t)$ are non-negative integers. Since $\mu_{ii}(t) = 0$, we have $\mu^r_{ii}(t) = \mu^f_{ii}(t) = 0$ for all $i \in \mathcal{N} \cup \{0\}$. All exogenous arrivals are considered as real packets, so knowing the value of $\mu_{0j}(t)$, we set $\mu^r_{0j}(t) = \mu_{0j}(t)$ and $\mu^f_{0j}(t) = 0$ for every $j \in \{1, \dots, N\}$. The real and fake parts of $\mu_{ij}(t)$ for $i \in \mathcal{N}, j \in \mathcal{N} \cup \{0\}$ can be set arbitrarily to satisfy:

$$b^r_n(t) = \sum_{j=0}^N \mu^r_{nj}(t), \quad b^f_n(t) = \sum_{j=0}^N \mu^f_{nj}(t) \qquad \forall n \in \mathcal{N}$$

$$\mu_{ij}(t) = \mu^r_{ij}(t) + \mu^f_{ij}(t) \qquad \forall i \in \mathcal{N}, j \in \mathcal{N} \cup \{0\}$$

Therefore, all $\mu^r_{ij}(t)$ and $\mu^f_{ij}(t)$ for all $i,j \in \mathcal{N} \cup \{0\}$ are fully determined. This is illustrated in Figure 3.

Let $a^r_n(t) = \sum_{i=0}^N \mu^r_{in}(t)$ and $a^f_n(t) = \sum_{i=1}^N \mu^f_{in}(t)$ be the aggregated real and fake parts of $a_n(t)$. They are fully determined and $a_n(t) = a^r_n(t) + a^f_n(t)$.

The floating-queue algorithm always admits a real packet to a real queue as much as allowed by its buffer space. Let $a^{r'}_n(t)$ denote the amount of packets in $a^r_n(t)$ that are admitted to the real queue n at time t. Another part of $a^r_n(t)$, which is dropped, is denoted by $d_n(t)$ and becomes fake packets. Let $a^{f'}_n(t)$ denote the admitted fake arrivals, including original fake arrivals $a^f_n(t)$ and dropped packets $d_n(t)$. The arrival dynamic of queue $n \in \mathcal{N}$ is

$$a^{r'}_n(t) = \min\left[B - Q^r_n(t), a^r_n(t)\right] \tag{14}$$

$$d_n(t) = a^r_n(t) - a^{r'}_n(t) \tag{15}$$

$$a^{f'}_n(t) = a^f_n(t) + d_n(t). \tag{16}$$

It is easy to see that $a^{r'}_n(t) + a^{f'}_n(t) = a^r_n(t) + a^f_n(t)$.

C. Real and Fake Queuing Dynamics

The dynamics of real and fake queues of standard queue $n \in \mathcal{N}$ are

$$Q^r_n(t+1) = Q^r_n(t) - b^r_n(t) + a^{r'}_n(t) \tag{17}$$

$$Q^f_n(t+1) = \max\left[Q^f_n(t) - b^f_n(t), 0\right] + a^{f'}_n(t). \tag{18}$$

Lemma 1: From any time t_0, when $Q(t_0) = Q^r(t_0) + Q^f(t_0)$, it follows for all $t \in \{t_0, t_0+1, t_0+2, \dots\}$ that

$$Q(t) = Q^r(t) + Q^f(t).$$

Proof: We prove this lemma by induction. For t_0, $Q(t_0) = Q^r(t_0) + Q^f(t_0)$ by the assumption. Suppose $Q(t) = Q^r(t) + Q^f(t)$ at time t. At queue $n \in \mathcal{N}$, we have

$$\begin{aligned} Q_n(t+1) &= \max\left[Q_n(t) - b_n(t), 0\right] + a_n(t) \\ &= \max\left[Q^r_n(t) + Q^f_n(t) - b^r_n(t) - b^f_n(t), 0\right] \\ &\quad + a^{r'}_n(t) + a^{f'}_n(t). \end{aligned} \tag{19}$$

```
**Initialization:** $Q^r(0) = \mathbf{0}$ and $Q^f(0)$
**for** $t \in \{0, 1, 2, \ldots\}$ **do**
    Observer $S(t)$ and let $Q(t) = Q^r(t) + Q^f(t)$.
    Choose $x(t)$ that solves (3).
    Calculate $(a_n(t), b_n(t))$ $\forall n \in \mathcal{N}$.
    Calculate $(b_n^r(t), b_n^f(t))$ as (12) and (13) $\forall n \in \mathcal{N}$.
    Adjust $(a_n^{r'}(t), a_n^{f'}(t))$ as (14)–(16) $\forall n \in \mathcal{N}$.
    $Q_n^r(t+1) = Q_n^r(t) - b_n^{r'}(t) + a_n^{r'}(t)$ $\forall n \in \mathcal{N}$.
    $Q_n^f(t+1) = \max\left[Q_n^f(t) - b_n^f(t), 0\right] + a_n^{f'}(t)$ $\forall n \in \mathcal{N}$.
**end for**
```

Fig. 4. The floating-queue algorithm

When there are not enough real packets, $Q_n^r(t) < b_n(t)$, it follows that $Q_n^r(t) - b_n^r(t) = 0$ from (12). Equation (19) becomes

$$\begin{aligned} Q_n(t+1) &= \max\left[Q_n^f(t) - b_n^f(t), 0\right] + a_n^{f'}(t) + a_n^{r'}(t) + 0 \\ &= Q_n^f(t+1) + Q_n^r(t+1). \end{aligned}$$

When there are enough real packets, $Q_n^r(t) \geq b_n(t)$, we have that $b_n^f(t) = 0$ from (12) and (13). Equation (19) becomes

$$\begin{aligned} Q_n(t+1) &= Q_n^f(t) + a_n^{f'}(t) + Q_n^r(t) - b_n^r(t) + a_n^{r'}(t) \\ &= Q_n^f(t+1) + Q_n^r(t+1). \end{aligned}$$

Thus, $Q_n(t+1) = Q_n^f(t) + Q_n^r(t)$ for all $n \in \mathcal{N}$. ■

The implication of Lemma 1 is that, although the floating-queue algorithm implements these real and fake queues instead of the standard queues, the dynamics of $Q(t)$ and $Q^r(t) + Q^f(t)$ are the same. Hence, when decision $x(t)$ is chosen by solving (3) with $Q^r(t) + Q^f(t)$ instead of $Q(t)$, all decisions $\{x(t)\}_{t=0}^\infty$ under the standard algorithm (in Figure 2) are identical to the decisions $\{x(t)\}_{t=0}^\infty$ under the floating-queue algorithm (in Figure 4), given that $Q^r(0) + Q^f(0) = Q(0)$. Yet, the buffer size of each real queue in the floating-queue algorithm is B. Let $Q^f(0) \in \mathbb{Z}_+^N$ be a pre-defined initial condition for the fake queues. The floating-queue algorithm is summarized in Figure 4.

We prove a useful lemma of the floating-queue algorithm, which will be used in Section V-B.

Lemma 2: Under the floating-queue algorithm, when the buffer size of the real queue $n \in \mathcal{N}$ is $B \geq 2\delta^{(\max)}$, if $d_n(t) > 0$, then $Q_n^f(t+1) > Q_n^f(t)$.

Proof: Event $d_n(t) > 0$ implies that $a_n^r(t) > a_n^{r'}(t)$ from (15) and $a_n^{r'}(t) = B - Q_n^r(t)$ from (14), so $Q_n^r(t) > B - a_n^r(t)$. When $B \geq 2\delta^{(\max)}$, we have $Q_n^r(t) > 2\delta^{(\max)} - a_n^r(t) \geq \delta^{(\max)}$, and there are enough real packets for all services. Therefore, all services take real packets and $b_n^r(t) = b_n(t)$ and $b_n^f(t) = 0$ from (12) and (13). From (18) and (16), we have

$$Q_n^f(t+1) = Q_n^f(t) + a_n^{f'}(t) = Q_n^f(t) + a_n^f(t) + d_n(t) > Q_n^f(t).$$

■

The interpretation of Lemma 2 is that, for any queue n with buffer size $B \geq 2\delta^{(\max)}$, if real packets are dropped at time t, then the fake backlogs at time $t+1$ always increase.

V. Performance Analysis

The steady-state performance of the floating-queue algorithm is analyzed by bounding from below the admitted real arrivals at each queue $n \in \mathcal{N}$. Define $\Theta_n(t) \triangleq$

$$\left(a_n^r(t), a_n^f(t), a_n^{r'}(t), a_n^{f'}(t), b_n^r(t), b_n^f(t), Q_n^r(t), Q_n^f(t)\right)$$

as a sample path of the arrivals, services, and backlogs of queue n that is generated by the floating-queue algorithm at time t. For any positive integer T and starting time t_0, a sample path of queue n from t_0 to $t_0 + T$ is denoted by $\{\Theta_n(t)\}_{t=t_0}^{t_0+T}$. Note that $Q_n(t)$ can be recovered from this sample path as $Q_n(t) = Q_n^r(t) + Q_n^f(t)$.

From sample path $\Theta_n(t)$, the amount of real arrivals are $a_n^r(t)$, and the amount of *admitted* real arrivals are $a_n^{r'}(t)$, which depend on the floating-queue mechanism (14). To lower bound this admitted real arrival $a_n^{r'}(t)$, we construct another mechanism, called "lower-bound policy", that operates over the sample path. It has a different rule for counting admitted real packets (later defined as $\hat{a}_n^r(t)$), which is part of the real arrivals $a_n^r(t)$. We will show (Lemma 6) that the amount of admitted real arrivals under the floating-queue algorithm is lower bounded by the amount of admitted real arrivals under the lower-bound policy. Using this lower bound, the performance of the floating-queue algorithm can be analyzed.

A. Lower-Bound Policy

In this section, queue $n \in \mathcal{N}$ is fixed and the lower-bound policy is defined for this queue. For simplicity, assume that the buffer size B is even and $B \geq 2\delta^{(\max)}$.

Recall that γ^{V*} is the Lagrange multiplier of problem (9), and $\delta^{(\max)}$ is the upper bound on $a_n(t)$ and $b_n(t)$. Define

$$\mathcal{B}_n \triangleq \left[\gamma_n^{V*} - B/2 + \delta^{(\max)}, \gamma_n^{V*} + B/2 - \delta^{(\max)}\right].$$

Let $\hat{a}_n^r(t)$ denote the number of admitted real packets under the lower-bound policy at time t. Given any sample path $\Theta_n(t)$ having real arrivals $a_n^r(t)$ and total backlogs $Q_n(t)$, the lower-bound policy counts real packets as

$$\hat{a}_n^r(t) = \begin{cases} a_n^r(t) & , Q_n(t) \in \mathcal{B}_n \\ 0 & , Q_n(t) \notin \mathcal{B}_n. \end{cases} \tag{20}$$

Let $\hat{d}_n(t)$ denote the number of dropped packets under the lower-bound policy at time t. It satisfies

$$\hat{d}_n(t) = a_n^r(t) - \hat{a}_n^r(t). \tag{21}$$

Notice that $\hat{a}_n^r(t)$ and $\hat{d}_n(t)$ are artificial numbers and are not real and fake packets in a real system. These values can be determined by $a_n^r(t)$ and $Q_n(t)$ of sample path $\Theta_n(t)$.

B. Sample Path Analysis

The goal of this section is to show (Lemma 6) that, for any sample path $\{\Theta_n(t)\}_{t=t_0}^\infty$ of queue $n \in \mathcal{N}$ and any positive integer T, the admitted real arrivals under the floating-queue algorithm with buffer size B is lower bounded by

$$\sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t) \geq \sum_{t=t_0}^{t_0+T-1} \hat{a}_n^r(t) - B.$$

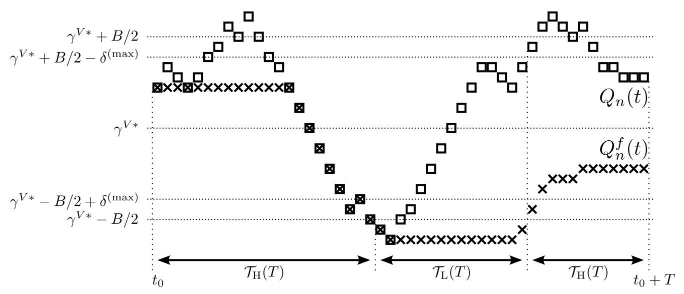

Fig. 5. Time interval $\mathcal{T}(T)$ is partitioned into $\mathcal{T}_{\rm H}(T)$ and $\mathcal{T}_{\rm L}(T)$.

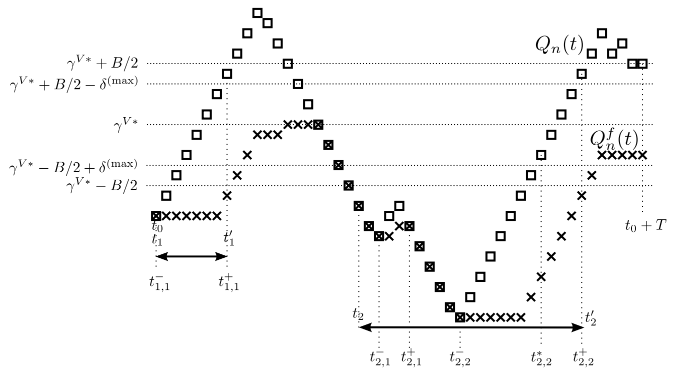

Fig. 6. Partitioning $\mathcal{T}_{\rm L}(T)$ into intervals of t_k, t'_k, $t^-_{k,j}$, and $t^+_{k,j}$

In this section, queue n is fixed and analyzed; however, the analysis results hold for every queue $n \in \mathcal{N}$.

For any starting time t_0 and any positive integer T, define $\mathcal{T}(T) \triangleq \{t_0, \dots, t_0+T\}$ as a time interval of consideration. It can be partitioned into disjoint sets $\mathcal{T}_{\rm H}(T)$ and $\mathcal{T}_{\rm L}(T)$, which are illustrated in Figure 5, where

$$\mathcal{T}_{\rm H}(T) \triangleq \{t \in \mathcal{T}(T) : Q_n^f(t) \geq \gamma_n^{V*} - B/2\}$$
$$\mathcal{T}_{\rm L}(T) \triangleq \{t \in \mathcal{T}(T) : Q_n^f(t) < \gamma_n^{V*} - B/2\}.$$

Time interval $\mathcal{T}_{\rm L}(T)$ can be partitioned into disjointed intervals of time that starts when the fake queue in the sample path satisfies $Q_n^f(t) < \gamma_n^{V*} - B/2$ and ends when it does not. This is illustrated in Figure 6. For $k \in \{1,2,\dots\}$, let

$$t_k = \operatorname{arginf}_{t \in \{t'_{k-1}+1,\dots,t_0+T\}} \left\{Q_n^f(t) < \gamma_n^{V*} - B/2\right\}$$
$$t'_k = \operatorname{arginf}_{t \in \{t_k+1,\dots,t_0+T\}} \left\{Q_n^f(t) \geq \gamma_n^{V*} - B/2\right\} - 1,$$

where $t'_0 = t_0 - 1$ and $\operatorname{arginf}_{t\in\{A,\dots,B\}}\{C(t)\} = B+1$ if $A > B$ or $C(t)$ is not satisfied for all $t \in \{A,\dots,B\}$. Let $K(T) = \operatorname{argmax}_{k \geq 0} \{t_k < t_0 + T + 1\}$ denote the number of intervals $\{t_k,\dots,t'_k\}$ contained in $\mathcal{T}(T)$.

When $K(T) > 0$, the time interval in interval $\{t_k,\dots,t'_k\}$ for $k \in \{1,2,\dots,K(T)\}$ can be partitioned into intervals between local minima and local maxima. Define $U(t) \triangleq \operatorname{arginf}_{\tau \in \{t+1,t+2,\dots\}} \left\{Q_n^f(\tau) > Q_n^f(t)\right\}$ to be the first time index after t that the fake queue increases. For $k \in \{1,2,\dots,K(T)\}, j \in \{1,2,\dots\}$, let

$$t^-_{k,j} = \min\Big[\operatorname{arginf}_{t \in \{t^+_{k,j-1}+1,\dots,t'_k\}} \{Q_n^f(t) < Q_n^f(t-1) \text{ and }$$
$$Q_n^f(t) \leq Q_n^f(\tau)\ \forall \tau \in \{t+1,\dots,U(t)\}\}, t'_k\Big]$$

$$t^+_{k,j} = \min\Big[\operatorname{arginf}_{t\in\{t^-_{k,j}+1,\dots,t'_k\}} \left\{Q_n^f(t) > Q_n^f(t+1)\right\}, t'_k\Big],$$

where $t^+_{k,0} = t_k - 1$ and $Q_n^f(t_0 - 1) = \infty$. Intuitively, during $\{t_k,\dots,t'_k\}$, $t^-_{k,j}$ is the first time index that the $j^{\rm th}$ local minimum is reached, and $t^+_{k,j}$ is the last time index of the $j^{\rm th}$ local maximum. This is illustrated in Figure 6. Let $J(k) = \operatorname{arginf}_{j>0}\left\{t^+_{k,j} = t'_k\right\}$ denote the number of local maxima during $\{t_k,\dots,t'_k\}$.

For a technical reason (used in Lemma 4), let $\mathcal{T}_{\rm A}(T) \triangleq \{(t_k - 1) \in \mathcal{T}(T) : k \in \{1,\dots,K(T)\}\}$. The following lemmas hold for the real arrivals in $\mathcal{T}_{\rm H}(T)\backslash\mathcal{T}_{\rm A}(T)$ and $\mathcal{T}_{\rm L}(T) \cup \mathcal{T}_{\rm A}(T)$.

Lemma 3: When $B \geq 2\delta^{(\max)}$, given any sample path $\{\Theta_n(t)\}_{t=t_0}^{t_0+T}$, the following relation holds

$$\sum_{t\in\mathcal{T}_{\rm H}(T)\backslash\mathcal{T}_{\rm A}(T)} a_n^{r'}(t) \geq \sum_{t\in\mathcal{T}_{\rm H}(T)\backslash\mathcal{T}_{\rm A}(T)} \hat{a}_n^r(t).$$

Proof: Two cases are examined.

1) When $Q_n(t) \in \mathcal{B}_n$ for any $t \in \mathcal{T}_{\rm H}(T)\backslash\mathcal{T}_{\rm A}(T)$, we have $a_n^{r'}(t) = a_n^r(t)$, because real queue n has enough buffer space:

$$\begin{aligned} Q_n^r(t) &= Q_n(t) - Q_n^f(t) \\ &\leq \left(\gamma_n^{V*} + B/2 - \delta^{(\max)}\right) - \left(\gamma_n^{V*} - B/2\right) \\ &\leq B - \delta^{(\max)}. \end{aligned}$$

The first inequality holds because of $Q_n(t) \in \mathcal{B}_n$ and $t \in \mathcal{T}_{\rm H}(T)$. For the lower-bound policy, we have $\hat{a}_n^r(t) = a_n^r(t)$, because $Q_n(t) \in \mathcal{B}_n$. So $a_n^{r'}(t) = \hat{a}_n^r(t)$.

2) When $Q_n(t) \notin \mathcal{B}_n$ for any $t \in \mathcal{T}_{\rm H}(T)\backslash\mathcal{T}_{\rm A}(T)$, we have $a_n^{r'}(t) \geq \hat{a}_n^r(t) = 0$, because $Q_n(t) \notin \mathcal{B}_n$ and $a_n^{r'}(t) \geq 0$.

These two cases implies the lemma. ■

Lemma 4: When $B \geq 2\delta^{(\max)}$, given sample path $\{\Theta_n(t)\}_{t=t_0}^{t_0+T}$ with $Q_n^r(t_0) = 0$, the following holds

$$\sum_{t\in\mathcal{T}_{\rm L}(T)\cup\mathcal{T}_{\rm A}(T)} a_n^{r'}(t) \geq \sum_{t\in\mathcal{T}_{\rm L}(T)\cup\mathcal{T}_{\rm A}(T)} \hat{a}_n^r(t).$$

Proof: For $k \in \{1,\dots,K(T)\}, j \in \{1,\dots,J(k)\}$, 3 cases are examined.

1) For real arrivals $a_n^r(t)$ during $t \in \left\{t_k - 1,\dots,t^-_{k,1} - 2\right\}$ (if exists), the fake backlogs $Q_n^f(t)$ is non-increasing by the definition of $t^-_{k,1}$. From Lemma 2, the non-increasing implies no packet drops and $a_n^{r'}(t) = a_n^r(t)$ for $t \in \left\{t_k - 1,\dots,t^-_{k,1} - 2\right\}$. Since $\hat{a}_n^r(t) \leq a_n^r(t)$, it follows that $\sum_{t=t_k-1}^{t^-_{k,1}-2} a_n^{r'}(t) \geq \sum_{t=t_k-1}^{t^-_{k,1}-2} \hat{a}_n^r(t)$. For a special case when $t_1 = t_0$, same argument can be used to obtain $\sum_{t=t_1}^{t^-_{1,1}-2} a_n^{r'}(t) \geq \sum_{t=t_1}^{t^-_{1,1}-2} \hat{a}_n^r(t)$.

2) For real arrivals $a_n^r(t)$ during $t \in \left\{t^-_{k,j} - 1,\dots,t^+_{k,j}\right\}$, it can be shown that $\sum_{t=t^-_{k,j}-1}^{t^+_{k,j}} a_n^{r'}(t) \geq \sum_{t=t^-_{k,j}-1}^{t^+_{k,j}} \hat{a}_n^r(t)$. This result is proven in [18].

3) For real arrivals $a_n^r(t)$ during $t \in \left\{t^+_{k,j} + 1,\dots,t^-_{k,j+1} - 2\right\}$ (if exists), the fake backlogs $Q_n^f(t)$ is non-increasing by the definitions of $t^+_{k,j}$ and $t^-_{k,j+1}$. Lemma 2 implies that $\sum_{t=t^+_{k,j}+1}^{t^-_{k,j+1}-2} a_n^{r'}(t) \geq \sum_{t=t^+_{k,j}+1}^{t^-_{k,j+1}-2} \hat{a}_n^r(t)$.

Since $Q_n^r(t_0) = 0$, the time interval $\{t_k, \dots, t_k'\}$ for $k \in \{1, \dots, K(T)\}$ starts with either the first case or the second case and ends with the second case. Thus, it is clear that, if real arrivals under the lower-bound policy are dominated over every subinterval, then they are also dominated over the union of these subintervals. ■

Lemma 5: When $B \geq 2\delta^{(\max)}$, given sample path $\{\Theta_n(t)\}_{t=t_0}^{\infty}$ with $Q_n^r(t_0) = 0$ and positive integer T, it holds that

$$\sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t) \geq \sum_{t=t_0}^{t_0+T-1} \hat{a}_n^r(t).$$

Proof: Disjoint time intervals $\mathcal{T}_{\text{H}}(T-1) \backslash \mathcal{T}_{\text{A}}(T-1)$ and $\mathcal{T}_{\text{L}}(T-1) \cup \mathcal{T}_{\text{A}}(T-1)$ are the partitions of $\mathcal{T}(T-1)$. Then Lemma 3 and Lemma 4 imply the lemma. ■

The above lemma is not general, since it requires $Q_n^r(t_0) = 0$. Now its general version is provided.

Lemma 6: When $B \geq 2\delta^{(\max)}$, given any sample paths $\{\Theta_n(t)\}_{t=t_0}^{\infty}$ and positive integer T, it holds for any $Q_n^r(t_0) \in \{0, 1, \dots, B\}$ that

$$\sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t) \geq \sum_{t=t_0}^{t_0+T-1} \hat{a}_n^r(t) - B.$$

Proof: Construct sample path $\left\{\tilde{\Theta}_n(t)\right\}_{t_{-1}}^{\infty}$ with $\tilde{\Theta}_n(t) = \left(\tilde{a}_n^r(t), \tilde{a}_n^f(t), \tilde{a}_n^{r'}(t), \tilde{a}_n^{f'}(t), \tilde{b}_n^r(t), \tilde{b}_n^f(t), \tilde{Q}_n^r(t), \tilde{Q}_n^f(t)\right)$ and

- $t_{-1} < t_0$,
- $\tilde{\Theta}_n(t) = \Theta_n(t)$ for all $t \in \{t_0, t_0+1, \dots\}$,
- $\tilde{Q}_n^f(t) = Q_n^f(t_0)$ for all $t \in \{t_{-1}, \dots, t_0 - 1\}$,
- $\tilde{Q}_n^r(t_{-1}) = 0$, $\tilde{Q}_n^r(t_0) = Q_n^r(t_0) = \sum_{t=t_{-1}}^{t_0-1} \tilde{a}_n^{r'}(t)$,
- $\tilde{a}_n^{f'}(t) = \tilde{b}_n^r(t) = \tilde{b}_n^f(t) = 0$ for all $t \in \{t_{-1}, \dots, t_0 - 1\}$.

The last two conditions automatically set the values of $\left\{\tilde{a}_n^r(t), \tilde{a}_n^f(t), \tilde{Q}_n^r(t)\right\}_{t_{-1}}^{t_0-1}$. This new sample path satisfies Lemma 5. Let $\hat{\tilde{a}}_n^r(t)$ denote the admitted real arrivals under the lower-bound policy of the new sample path. Since $\sum_{t=t_0}^{t_0+T-1} \tilde{a}_n^{r'}(t) = \sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t)$, it follows that

$$\sum_{t=t_{-1}}^{t_0-1} \tilde{a}_n^{r'}(t) + \sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t) = \sum_{t=t_{-1}}^{t_0-1} \tilde{a}_n^{r'}(t) + \sum_{t=t_0}^{t_0+T-1} \tilde{a}_n^{r'}(t)$$
$$\geq \sum_{t=t_{-1}}^{t_0-1} \hat{\tilde{a}}_n^r(t) + \sum_{t=t_0}^{t_0+T-1} \hat{\tilde{a}}_n^r(t) = \sum_{t=t_{-1}}^{t_0-1} \hat{\tilde{a}}_n^r(t) + \sum_{t=t_0}^{t_0+T-1} \hat{a}_n^r(t).$$

The first inequality is the application of Lemma 5, and the last equality holds, because $\{Q_n(t)\}_{t=t_0}^{\infty}$ of both original and new sample paths are identical.

Therefore, we have

$$\sum_{t=t_0}^{t_0+T-1} a_n^{r'}(t) \geq \sum_{t=t_0}^{t_0+T-1} \hat{a}_n^r(t) - B.$$

The inequality uses the facts that $\sum_{t=t_{-1}}^{t_0-1} \hat{\tilde{a}}_n^r(t) \geq 0$ and $\sum_{t=t_{-1}}^{t_0-1} \tilde{a}_n^{r'}(t) = Q_n^r(t_0) \leq B$. ■

C. Performance of Floating-Queue Algorithm

1) Average Drops: The average drops at each queue is analyzed using the steady state and sample path results. Recall that constants D, K, c^* are defined in Theorem 1.

Lemma 7: Suppose $B > 2(\delta^{(\max)} + D)$. In the steady state, the average drops at real queue $n \in \mathcal{N}$ under the floating-queue algorithm is bounded by

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[d_n(t)] \leq \delta^{(\max)} c^* e^{\frac{-[B/2-\delta^{(\max)}-D]}{K}}.$$

Proof: We consider queue $n \in \mathcal{N}$. Let $\mathbb{I}\{A\}$ be an indicator function of statement A such that $\mathbb{I}\{A\} = 1$ if statement A is true; otherwise $\mathbb{I}\{A\} = 0$. Equation (20) can be written as $\hat{a}_n^r(t) = a_n^r(t)\mathbb{I}\{Q_n(t) \in \mathcal{B}_n\} = a_n^r(t) - a_n^r(t)\mathbb{I}\{Q_n(t) \notin \mathcal{B}_n\}$. Then we have that

$$\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[\hat{a}_n^r(t)]$$
$$= \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[a_n^r(t) - a_n^r(t)\mathbb{I}\{Q_n(t) \notin \mathcal{B}_n\}]$$

Dividing the result in Lemma 6 by T and taking an expectation yields

$$\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n^{r'}(t)\right] \geq \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[\hat{a}_n^r(t)] - \frac{B}{T}.$$

Combining the above two relations gives

$$\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n^{r'}(t)\right] \geq \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[a_n^r(t)] - \frac{B}{T}$$
$$- \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[a_n^r(t)\mathbb{I}\{Q_n(t) \notin \mathcal{B}_n\}].$$

It follow from (15) that

$$\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[d_n(t)] \leq \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \delta^{(\max)} \mathbb{P}\{Q_n(t) \notin \mathcal{B}_n\} + \frac{B}{T}$$

Taking limit as T approaches infinity yields

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}[d_n(t)]$$
$$\leq \delta^{(\max)} \lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{P}\{Q_n(t) \notin \mathcal{B}_n\}. \quad (22)$$

In steady state, Theorem 1 with $\beta = \frac{B/2-\delta^{(\max)}-D}{K}$ yields

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{P}\{Q_n(t) \notin \mathcal{B}_n\}$$
$$\leq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{P}\left\{\exists n, \left|Q_n(t) - \gamma_n^{V*}\right| > B/2 - \delta^{(\max)}\right\}$$
$$= \mathcal{P}(D, B/2 - \delta^{(\max)} - D) \leq c^* e^{-[B/2-\delta^{(\max)}-D]/K}.$$

Applying the above bound to (22) proves the lemma. ■

2) Delay: At each queue, the average delay experienced by real packets is derived by invoking Little's law [17]. Define

$$\bar{a}_n^r \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n^r(t)\right],\ \bar{a}_n \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n(t)\right].$$

Lemma 8: Suppose $B > 2(\delta^{(\max)} + D)$. In the steady state, the average delay at real queue $n \in \mathcal{N}$ under the floating-queue algorithm is bounded by

$$\text{Per-hop delay} \le \frac{B}{\bar{a}_n^r - \delta^{(\max)} c^* e^{-[B/2-\delta^{(\max)}-D]/K}}.$$

Proof: Since the buffer size of queue $n \in \mathcal{N}$ is B, Little's law implies:

$$\begin{aligned}\text{Per-hop Delay} &= B \Big/ \left[\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n^{r'}(t)\right]\right] \\ &= B \Big/ \left[\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[a_n^r(t) - d_n(t)\right]\right] \\ &\le B \Big/ \left[\bar{a}_n^r - \delta^{(\max)} c^* e^{-[B/2-\delta^{(\max)}-D]/K}\right].\end{aligned}$$

■

The implication of Lemma 8 is that, when B is large enough such that $\delta^{(\max)} c^* e^{-[B/2-\delta^{(\max)}-D]/K}$ and the number of drops at other queues are negligible, $\bar{a}_n^r$ is approximately $\bar{a}_n$, and the average delay is $O(B)$.

3) Objective Cost: The average objective cost is considered in two cases. Let $f^{(\text{FQ})}(t)$ denote the cost under the floating-queue algorithm at time t, and $f^{(\text{FQ})} \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[f^{(\text{FQ})}(t)\right]$ denote the expected time-average cost under the floating-queue algorithm.

Drop-Independent Cost:
In this case, packet drops do not affect the objective cost. Such cost can be the energy expenditure that is spent to transmit both real and fake packets. Due to this independence, the average cost follows immediately from the result of drift-plus-penalty policy (4).

Theorem 2: Suppose each real queue has buffer size $B > 2(\delta^{(\max)} + D)$. When $V > 0$ and packet drops do not incur any penalty cost, the floating-queue algorithm achieves:

$$\begin{aligned} f^{(\text{FQ})} = f^{(\text{DPP})} &\le f^{(\text{opt})} + O(1/V) \\ \text{Per-hop delay} &\le O(B/(1-e^{-B})) = O(B) \\ \text{Average drops} &\le O(e^{-B}). \end{aligned}$$

It can be shown that the transient time of the drift-plus-penalty algorithm is $O(V)$, so parameter V cannot be set to infinity.

Drop-Dependent Cost:
In this case, packet drops affect the objective cost. Such cost can be the amount of admitted packets. Let $\kappa < \infty$ be a maximum penalty cost per one unit of packet drop. Then we have the following result.

Theorem 3: Suppose $B > 2(\delta^{(\max)} + D)$. When $V > 0$ and κ is a maximum penalty cost per one unit of packet drop, the floating-queue algorithm achieves:

$$\begin{aligned} f^{(\text{FQ})} &\le f^{(\text{opt})} + O(1/V) + O(e^{-B}) \\ \text{Per-hop delay} &\le O(B/(1-e^{-B})) = O(B) \\ \text{Average drops} &\le O(e^{-B}) \end{aligned}$$

Proof: Recall that $f(t)$ is a cost incurred at time t under the drift-plus-penalty policy. At each time t, we have

$$f^{(\text{FQ})}(t) \le f(t) + \kappa \sum_{n=1}^{N} d_n(t).$$

Summing from t_0 to $t_0 + T - 1$, dividing by T, and taking an expectation gives

$$\begin{aligned}\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[f^{(\text{FQ})}(t)\right] \le \frac{1}{T} &\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[f(t)\right] \\ &+ \frac{\kappa}{T} \sum_{t=t_0}^{t_0+T-1} \sum_{n=1}^{N} \mathbb{E}\left[d_n(t)\right].\end{aligned}$$

Taking a limit as T approaches infinity gives

$$\begin{aligned}&\lim_{T\to\infty} \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[f^{(\text{FQ})}(t)\right] \\ &\le f^{(\text{DPP})} + \kappa N \delta^{(\max)} c^* e^{-[B/2-\delta^{(\max)}-D]/K} \\ &\le f^{(\text{opt})} + O(1/V) + O(e^{-B})\end{aligned}$$

■

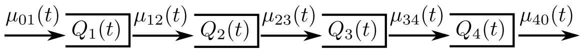

Fig. 7. Line network

VI. Simulation

A line network with 4 queues, shown in Figure 7, is simulated in two scenarios. The common network configuration is as follows. In each time slot, an exogenous packet arrives with probability 0.92. Transmission $\mu_{ij}(t)$ is orthogonal and depends on channel state that is "good" with probability 0.9 and "bad" with probability 0.1 for $(i,j) \in \{(1,2),(2,3),(3,4),(4,0)\}$.

A. Power Minimization

In this scenario, all exogenous arrivals are admitted. When channel state is "good", one packet is transmitted using 1 unit of power; otherwise 2 units of power are used. The goal is to stabilize this network while minimizing the power usage. Note that the optimal average minimum power is $1\times 0.9 + 2\times 0.02 = 0.94$ per hop, and the average total power is 3.76.

Simulation results of this scenario are shown Figure 8. The time average power expenditure is 3.761 for all buffer sizes B. In Figure 8, the average delay increases linearly with the buffer size, and the average drops decrease exponentially with the buffer size. This result confirms the bounds in Theorem 2.

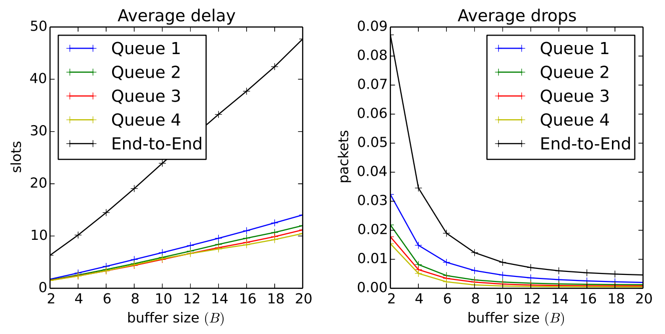

Fig. 8. Results of power minimization problem with $V = 200$

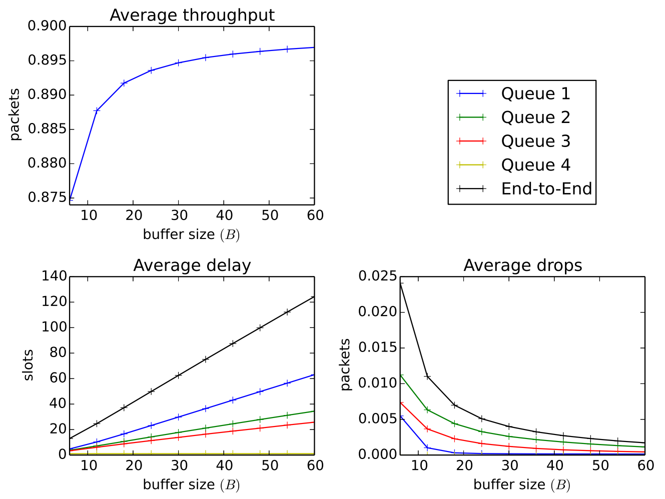

Fig. 9. Results of throughput maximization problem with $V = 200$

B. Throughput Maximization

In this scenario, a network decides to admit random exogenous arrival in each time slot. The goal is to maximize the time-average end-to-end throughput, which are real packets. Packet drops reduce the value of this objective function. Transmission $\mu_{ij}(t) = 1$ is possible if its channel state is "good"; otherwise the transmission is not allowed. Note that the maximum admission rate is 0.9, because of the limitation of the average transmission rate. Figure 9 shows the simulation results of this scenario, which comply with the bounds in Theorem 3.

VII. Conclusion

We propose the general floating-queue algorithm that allows the stochastic network optimization framework to operate with finite buffers. When the buffer size at each queue is B, we prove the proposed algorithm achieves within $O(e^{-B})$ of optimal utility, while the average per-hop delay is $O(B)$. The finiteness incurs $O(e^{-B})$ drops, decreasing exponentially. We confirm the theoretical results with simulations.

References

[1] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, 2010.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, Dec. 1992.

[3] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 6, Dec. 2007.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, Jan. 2005.

[5] S. Supittayapornpong and M. J. Neely, "Quality of information maximization for wireless networks via a fully separable quadratic policy," *Networking, IEEE/ACM Transactions on*, 2014, to appear.

[6] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, Aug. 2006.

[7] ——, "Optimal energy and delay tradeoffs for multiuser wireless downlinks," *Information Theory, IEEE Transactions on*, vol. 53, no. 9, Sep. 2007.

[8] L. Huang and M. J. Neely, "Delay reduction via lagrange multipliers in stochastic network optimization," *Automatic Control, IEEE Transactions on*, vol. 56, no. 4, Apr. 2011.

[9] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near-optimal utility-delay tradeoff," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 3, Jun. 2013.

[10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '10. ACM, 2010.

[11] J. Ryu, V. Bhargava, N. Paine, and S. Shakkottai, "Back-pressure routing and rate control for icns," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. ACM, 2010.

[12] E. Athanasopoulou, L. Bui, T. Ji, R. Srikant, and A. Stolyar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *Networking, IEEE/ACM Transactions on*, Feb. 2013.

[13] B. Ji, C. Joo, and N. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, Oct. 2013.

[14] L. B. Le, E. Modiano, and N. Shroff, "Optimal control of wireless networks with finite buffers," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 4, Aug. 2012.

[15] D. Xue, R. Murawski, and E. Ekici, "Distributed utility-optimal scheduling with finite buffers," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, May 2012.

[16] D. Bertsekas, A. Nedić, and A. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[17] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[18] S. Supittayapornpong and M. J. Neely, "Achieving utility-delay-reliability tradeoff in stochastic network optimization with finite buffers," *arXiv:1501.03457 [math.OC]*, Jan. 2015.