AFRL-RI-RS-TR-2021-129

TAMING DELAY AND CONVERGENCE SPEED IN TACTICAL AUTONOMOUS SWARMS NETWORK OPTIMIZATION

IOWA STATE UNIVERSITY

JULY 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2021-129 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
ELIZABETH S. BENTLEY
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor,
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JULY 2021	FINAL TECHNICAL REPORT	JULY 2018 – JAN 2021

4. TITLE AND SUBTITLE

TAMING DELAY AND CONVERGENCE SPEED IN TACTICAL AUTONOMOUS SWARMS NETWORK OPTIMIZATION

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-18-1-0107

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Jia Liu

5d. PROJECT NUMBER
T2PD

5e. TASK NUMBER
IO

5f. WORK UNIT NUMBER
UN

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Iowa State University
206 Atanasoff Hall
Ames IA 50011

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITGB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2021-129

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this proposed research program, our overarching goal is to develop an analytical foundation for low-delay and fast-converging distributed network control and optimization based on second-order information (SOI). Specifically, in this proposed research program, we will focus on the EMANE-based simulation development of momentum-based joint congestion control and multi-path routing. Our approach is based on the key idea of separating the queue-lengths from the weights in the back-pressure-based approach, and then uses a weight-updating scheme that utilizes only one more time-slot of memory of the weight change history.

15. SUBJECT TERMS

Stochastic network optimization, autonomous swarms

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**ELIZABETH S. BENTLEY**
U	U	U	UU	36	19b. TELEPHONE NUMBER (*Include area code*) **(315) 330-2371**

Table of Contents

Table of Figures

Summary

This research program addresses the challenge of distributed control and optimization for next generation autonomous swarm network systems, where the rapidly changing and super-dynamic network states (e.g., network topologies, spectrum and channel state information, data buffer queueing states, etc.) necessitate fast-convergence and low-delay in distributed optimization algorithms. Built upon the PI's recent research on network control and optimization that leverages *second-order information* (SOI), in this research program, we propose a series of new distributed algorithmic techniques that offer *orders of magnitudes improvements* in both convergence speed and queueing delay compared to the traditional approaches, while attaining the same provable network-utility optimality.

Specifically, our research tasks in this project are focused on the EMANE emulation implementations of momentum-based (Heavy-ball) joint congestion control and multi-path routing (partial SOI). Our proposed research program takes an integrated and holistic approach that draws techniques from areas of mathematical modeling, optimization theory, control theory, queueing theory, and stochastic analysis. The proposed research will not only advance our knowledge in the algorithmic design for next generation complex networks, but will also serve a critical need in the general networking research community by exploring new frontiers in SOI-based network control and optimization.

The proposed methodologies will impact a broad range of applications such as airborne networks and UAS imagery/video, and particularly in systems where control and optimization actions could not afford large delay and slow convergence. Substantial collaborations with AFRL will be pursued to foster potential transition avenues for this research effort.

1 Introduction

Background and Motivation: With the proliferation of communication networks being deployed on battlefields and the large amount of mobile data they generated, today's wireless network technologies are being stretched to their limits. Not only does the explosive growth of tactical information call for an ever-increasing network capacity, the complex coordinations for large-scale wireless networks also introduce stringent **latency** and **convergence speed** requirements in real-time control and optimization. To design highly efficient optimization algorithms to cope with the emerging tactical wireless networks, a key aspect is to *efficiently* deal with the cross-interactions between congestion control and link scheduling, both within and across protocol stack layers. As a result, recent years have witnessed a compelling need for low-latency and fast-converging joint congestion control and scheduling algorithms for tactical wireless networks. Moreover, joint congestion control and routing optimization is not only a requirement in information network design, but also a fundamental problem that lies at the heart of many complex network operations, such as smart grid demand response [1–3], supply chain management [4–7], transportation network traffic control [8, 9], just to name a few.

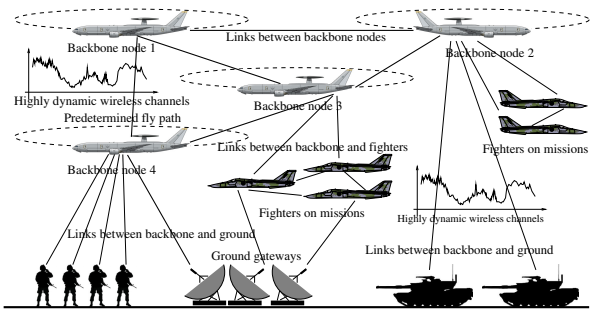


Figure 1: UAS communication with tight latency requirements under highly dynamic wireless networks.

A Motivating Example: To motivate the importance of fast-convergence, low-delay, and distributed design, here we use a UAS network as an example. A big challenge in controlling and optimizing a UAS network stems from the *fast-changing* and *highly-dynamic* network states (e.g., network topologies, spectrum/channel states, data buffer queueing states, etc.), which makes traditional congestion control, routing, and spectrum access techniques ineffective (see Figure 1 for an illustrative example). Such a highly dynamic nature necessitates **fast-convergence** of the network control and optimization algorithms. Otherwise, upon the completion of a slow convergence process, the network topology, spectrum/channel state information, and queueing states are most likely significantly altered, rendering all computational results and control actions outdated and useless.

Exacerbating the network control problem is the fact that the control actions correlate strongly to times when *large* amounts of data need to be transferred with *real-time* requirements (e.g., UAS imagery or video surveillance, etc.). Hence, when data arrivals surge, **low-delay** network control algorithms are needed to avoid excessive delay and large amounts of packet dropping (due to time-out events). Otherwise, a sudden large-scale network outage may occur, which could lead to not only wide-spread inconvenience, but also devastating battle defeats or even loss of lives. Moreover, the large geographical scale of airborne networks, heterogeneity of physical layer technologies between network subsystems, and the rapid response time requirements imply that the control and optimization algorithms can neither be centralized nor have high complexity. This requires the development of fully **distributed** algorithms that circumvent single-point-of-failure problems, are simple and easy to implement, and yet achieve provable optimality performance. □

Current State-of-the-Art: Due to the rapidly increasing mobile data demands, recent years have witnessed a large body of works on resource allocation that aim to maximize the network utility in wireless networks (see, e.g., [10–13], and [14] for a survey). This has led to an elegant mathematical decomposition framework, from which "loosely-coupled" congestion control, scheduling, and routing algorithms naturally emerge. These algorithms do not require statistical knowledge of either the arrivals or channel states. Instead, they only rely on queue-lengths and channel state information to make control decisions. These algorithms are also inherently connected to the Lagrangian dual decomposition framework plus the subgradient method in nonlinear optimization theory [10, 11], where (scaled) queue-lengths can be interpreted as Lagrangian dual variables and the queue-length updates play the role of subgradient directions.

Despite the attractive features of these queue-length-based algorithms (QLA), they suffer from several key limitations. First, in the existing QLA framework, it has been shown that a utility-optimality gap $O(1/K)$ can be achieved with an $O(K)$ penalty in queueing delay, where $K > 0$ is a system parameter. Hence, a small utility-optimality gap necessitates a large K and results in large queueing delay. To address this limitation, there have been significant efforts (e.g., [13, 15–17], etc.) in recent years focusing on reducing the queueing delay of these schemes (more in-depth discussions on related work later). Also, in the existing QLA framework, the queue-length-based weight adjustment ignores the curvature of the objective function contour and uses a small step-size in each iteration [10–13], which leads to unsatisfactory convergence speed. To address this problem, some second-order congestion control and routing/scheduling algorithms have recently been proposed to increase the convergence speed (see, e.g., [18, 19]). However, due to their complex algorithmic structures, these second-order approaches require a much larger information exchange overhead and do not scale well with the network size. These limitations of the existing approaches motivate us to pursue a new *heavy-ball* design in this project.

More specifically, in this project, we develop a heavy-ball-based weight adjustment scheme to dramatically reduce the queue-lengths and increase the convergence speed without impacting the network utility performance and without adding any computational complexity. Our approach is based on a clever idea of separating the queue-lengths from the weights, and then uses a weight updating scheme that utilizes *only one more slot of memory* of the weight change in the previous time-slot. Surprisingly, we show that this simple scheme offers *two control degrees of freedom* that allow us to achieve utility-optimality, low-delay, as well as fast-convergence.

Historically, the heavy-ball method was first proposed by Polyak in the 1960s [20] for solving unconstrained convex optimization problems, with the original goal of accelerating the convergence of the gradient descent method. The basic idea of the heavy-ball method is that, rather than using only the (sub)gradient information at the current iterate and being completely memoryless of the trajectory of past iterates, one computes the search direction using a linear combination of the current gradient (analogous to "potential") and the update direction in the previous step (analogous to "momentum"). The method was motivated by and can be viewed as a discrete version of the second-order ordinary differential equation (ODE) in physics that describes a heavy body's motion in a potential field, hence the name "heavy-ball." It has been shown in [21] that, by appropriately weighing the current "potential" and "momentum," the algorithm is insensitive to the objective contour, which leads to a much faster convergence. Indeed, the convergence acceleration advantage is our first fundamental rationale behind adopting the heavy-ball approach in wireless network cross-layer optimization. But surprisingly, our subsequent studies show that the benefits of adopting the heavy-ball idea go *far beyond* convergence acceleration.

We note, however, that due to a number of technical challenges, developing a heavy-ball-based solution to the utility maximization problem in wireless networks is not straightforward. First, since the heavy-ball method was originally designed for unconstrained static optimization problems, it is unclear how to modify the heavy-ball method for wireless network utility maximization, which is a constrained stochastic optimization problem with a much more complex problem structure. Second, unlike the obvious connection between queue-length and Lagrangian dual variables in the QLA design, the relationship between the heavy-ball method and the observable network state information (e.g., queue-lengths, channel states, etc.) is unknown. Hence, the trade-off between delay and network utility under a heavy-ball approach remains an open problem. Third, due to the inclusion of past iterations' values, the algorithmic structure of a heavy-ball method is different from that of the QLA approaches. As a result, conventional techniques used in QLA for establishing throughput-optimality and utility-delay tradeoff are not applicable. Thus, new analytical techniques are required in the performance analysis of the heavy-ball approach.

Technical Contributions: The main contribution of this project is that, for the first time, we develop a heavy-ball-based wireless network utility optimization framework that overcomes the aforementioned technical challenges. We establish a collection of new analytical results on dramatic delay reduction and fast convergence, while retaining utility-optimality. The main results and technical contributions of this paper are as follows:

- Motivated by the heavy-ball idea, we propose a new weight adjustment scheme for joint congestion control and routing/scheduling in wireless networks. Our work not only provides a synergy between the heavy-ball algorithm and observable network state information (queue-lengths and channel states) to allow simple implementation in practice, it also extends and generalizes the classical heavy-ball method from unconstrained static optimization to the constrained stochastic network utility optimization paradigm, thus advancing the state-of-the-art of the heavy-ball method in mathematical optimization theory.

- Under our heavy-ball-based joint congestion control and scheduling scheme with a β-parameterized momentum ($\beta \in [0, 1)$ is a system parameter typically chosen close to 1), we show that the delay is $(1 - \beta)$–*fraction* of that of the QLA approach. More specifically, our theoretical analysis unveils that a utility-optimality gap $O(1/K)$ can be achieved with an $O((1-\beta)K)+O((1+\beta)\sqrt{K})$ cost in queueing-delay, where the parameter K is inversely proportional to the step-size in the heavy-ball method. Further, in the asymptotic regime of K where β is chosen as $\beta = 1 - O(1/\sqrt{K})$, our heavy-ball algorithm achieves an $[O(1/K), O(\sqrt{K})]$ utility-delay trade-off, which is significantly better than the well-known $[O(1/K), O(K)]$ trade-off of the QLA methods.

- Given parameters K and β, we show that the convergence time of our heavy-ball-based algorithm scales as $O[\log(\sqrt{K})\ (-\log^{-1}(1 + \beta - \sqrt{\beta}))]$. Combined with the results in the previous bullet, our proposed heavy-ball algorithm offers an important and elegant *three-way trade-off* relationship governed by *two* control knobs in K and β. Most notably, *simultaneous* utility-optimality and low-delay is made possible by trading off convergence speed. We note that this important three-way trade-off relationship has *not* been discovered so far in the literature.

- In addition to the theoretical results, a **key focus** of this project is to develop high-fidelity EMANE-based simulations to test and verify our theoretical results and insights developed above. In this project, we have successfully developed a *Shim-layer-based* EMANE cross-layer simulation

platform to test our HeavyBall algorithms. Our EMANE-based simulation results show that all theoretical predictions are observable in high-fidelity simulations. Moreover, it is worth mentioning that our EMANE-based cross-layer simulation platform is highly versatile and could be of independent interests for other EMANE-based wireless network cross-layer simulations that are important to AFRL.

Related Work: In this section, we first review the state-of-the-art of the QLA literature that is closely related to this paper. As mentioned earlier, there have been significant efforts on reducing the delay of the QLA approaches. For example, in [13], a virtual queue technique similar to those in [22–24] was adopted, where the virtual queue-lengths evolve based on service rates that are a fraction of the actual service rates. In [16], a virtual backlog mechanism with place-holder bits instead of real data was proposed. It was shown that, by accepting some non-zero packet dropping probability, this approach achieves an $[O(1/K), O(\log^2(K))]$ utility-delay trade-off. An exponential Lyapunov virtual backlog method combined with a threshold-based packeting-dropping scheme was also proposed in [15] to achieve an $O(\log(K))$ delay. Although having a log-type delay scaling, a major limitation of [15, 16] is that choosing the size of place-holder bits in [16] and the threshold value in [15] all require *non-causal* global arrival and channel statistics (cf. [15, Eq. (17)], [16, Eq. (45)]), which is usually infeasible. Also, if the parameters are not set appropriately, these schemes may result in non-negligible packet dropping probability. To address this problem, a per-iteration learning was proposed in [17] to learn the optimal size of place-holder bits in an online fashion. However, the per-iteration learning component significantly increases the complexity of the algorithm. In some sense, all these delay reduction schemes can be viewed as sacrificing some throughput-optimality (reflected in reduced service rates or packet dropping) for delay reduction. In contrast, *without sacrificing any throughput-optimality and without requiring any non-causal statistical knowledge*, our heavy-ball scheme achieves an $[O(1/K), O(\sqrt{K})]$ utility-delay trade-off by setting $\beta = 1 - O(\frac{1}{\sqrt{K}})$. Moreover, our heavy-ball algorithm enables an elegant three-way trade-off that *cannot* be offered by the existing works in [13, 15–17].

Next, we provide further background of the heavy-ball method, and then review the related works in the heavy-ball domain. In the optimization literature, the heavy-ball method is also referred to as the multi-step or momentum method. Since its inception [20], the heavy-ball method has found applications in signal processing and machine learning (see, e.g., [25] and references therein). However, the heavy-ball method remains largely unexplored in networking research so far. To our knowledge, the only application of the heavy-ball method in networking areas can be found in [26], where the authors developed a heavy-ball-based scheme for Internet congestion control. We note that our work differs from [26] in the following key aspects: First, our proposed heavy-ball algorithm is a dynamic scheme that works with stochastic wireless channels, while the algorithm proposed in [26] solves a static congestion control rate optimization problem for wireline networks. Second, the algorithm in [26] requires some assumptions (c.f. [26, Sec. VII-C]) to turn the problem into an *unconstrained* formulation, so that the classical heavy-ball method can be applied. However, as indicated in [26], these assumptions restricted the use of the heavy-ball method to problems only with certain routing structures. In contrast, our proposed method can handle all network constraints and works with all utility optimization problems. Third, we derive explicit utility-delay-convergence trade-off scaling laws in this paper, while no such results were provided in [26].

Report Organization: Collectively, our results contribute to an exciting new design paradigm

in cross-layer network control and optimization theory that leverages momentum/memory information. The remainder of this report is organized as follows. Section 2 presents our proposed heavy-ball algorithm and the performance analysis of the proposed algorithm. Section 3 presents numerical results and Section 4 concludes this paper.

2 Methodology, Assumptions, and Procedures

In this section, we first present our network model and problem formulation in Section 2-A. Then, we present our heavy-ball-based network utility optimization algorithm and the main theoretical results in Section 2-B and Section 2-C, respectively.

2-A Network Model and Problem Formulation

In this section, we will consider a single-hop wireless network with N links, which can be used to represent a cellular base station (or access point) downlink/uplink channel with N users or a set of distributed communication pairs in an ad hoc network. We will later discuss how to extend the results to multi-hop wireless networks. The rationale behind this presentation approach is that the single-hop network model will allow us to present the *core idea* behind the heavy-ball-based design with less notational clutter, before we integrate further system dynamics in multi-hop wireless networks. Also, as mentioned above, since the single-hop model encompasses a large number of networks in practice, it is important in its own right.

Notation: We use boldface to denote matrices/vectors. We let $\mathbf{A}^\top$ denote the transpose of $\mathbf{A}$. We let $(\mathbf{A})_{ij}$ represent the entry in the i-th row and j-th column of $\mathbf{A}$ and let $(\mathbf{v})_m$ represent the m-th entry of $\mathbf{v}$. We let $\mathbf{I}_N$ and $\mathbf{O}_N$ denote the $N \times N$ identity and all-zero matrices, respectively. Also, we let $\mathbf{1}_N$ and $\mathbf{0}_N$ denote the N-dimensional all-one and all-zero vectors, respectively. We will often omit "N" for brevity if the dimension is clear from the context. We use $\|\cdot\|$ and $\|\cdot\|_1$ to denote L^2- and L^1-norms, respectively. We let $\lambda_{\min}\{\mathbf{A}\}$ and $\lambda_{\max}\{\mathbf{A}\}$ denote the smallest and largest eigenvalues of $\mathbf{A}$, respectively.

Network model: In the single-hop case, we will base our discussions on the cellular downlink system, as shown in Fig. 2. We assume that time is slotted and indexed by $t \in \{0, 1, 2, \ldots\}$. The channel between the base station and the receivers can be characterized by a total of M states and denoted by a matrix $\mathbf{\Pi} = [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_M] \in \mathbb{R}^{N\times M}$, where each column vector $\boldsymbol{\pi}_m \in \mathbb{R}^N$ corresponds to the N links' channel qualities under state m. For each $\boldsymbol{\pi}_m$, we let $\mathcal{C}_m$ denote the achievable rate region, which is defined as the convex hull of the feasible scheduling rate vectors, i.e., $\mathcal{C}_m \triangleq Conv\{x_1^{(m)}, \ldots, x_N^{(m)}\}$, where $Conv\{\cdot\}$ represents the convex hull operation and $x_n^{(m)}$ denotes a feasible rate of link n that can be scheduled under channel state m. We assume that, for each link n and channel state m, the feasible rates satisfy $x_n^{(m)} \leq s^{\max} < \infty$. We use a vector $\mathbf{x}^{(m)} = [x_1^{(m)}, \ldots, x_N^{(m)}]^\top \in \mathbb{R}^N$ to denote the feasible rates under channel state m.

We assume that the channel state process is independent and identically distributed in each time slot[1]. We let $\boldsymbol{\pi}[t]$ denote the channel state vector in time-slot t and let $p_m \triangleq \Pr\{\boldsymbol{\pi}[t] = \boldsymbol{\pi}_m\}$

[1]Following the same arguments such as those in [16, 27], our results can be readily generalized to Markov channel state processes.

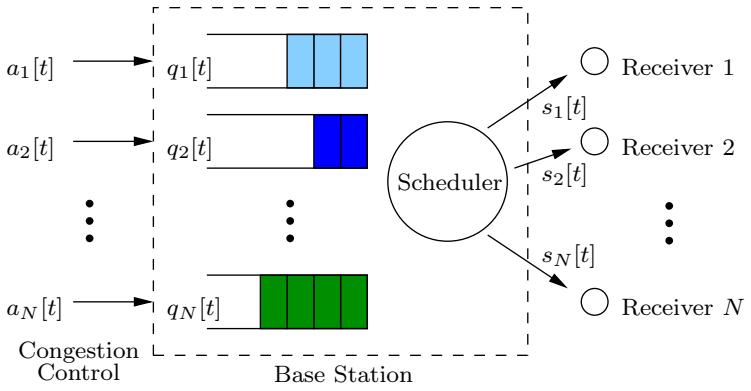


Figure 2: An illustration of the single-hop cellular downlink.

be the stationary distribution of the channel state process being in state m. We let $\bar{\mathcal{C}}$ represent the mean achievable rate region, which can be computed as:

$$\bar{\mathcal{C}} \triangleq \left\{ \mathbf{x} \left| \mathbf{x} = \sum_{m=1}^{M} p_m \mathbf{x}^{(m)}, \ \forall \mathbf{x}^{(m)} \in \mathcal{C}_m \right. \right\}.$$

Note that, in this project, neither the channel state statistics nor $\bar{\mathcal{C}}$ is assumed to be known at the base station.

Queue-stability: In each time-slot t, the controller observes the current channel state $\boldsymbol{\pi}[t] \in \mathbf{\Pi}$ and then chooses a service rate vector $\mathbf{s}[t] \triangleq [s_1[t], \ldots, s_N[t]]^\top \in \mathcal{C}_{\boldsymbol{\pi}[t]}$ and a congestion controlled rate vector $\mathbf{a}[t] \triangleq [a_1[t], \ldots, a_N[t]]^\top \in \mathbb{R}_+^N$. We assume that each link n is associated with a queue, whose queue-length in time-slot t is denoted as $q_n[t]$. Then, the queue-lengths evolve as:

$$q_n[t+1] = (q_n[t] - s_n[t] + a_n[t])^+, \quad \forall n, \tag{1}$$

where $(\cdot)^+ \triangleq \max\{0, \cdot\}$. Let $\mathbf{q}[t] \triangleq [q_1[t], \ldots, q_N[t]]^\top$ be the queue-length vector in time-slot t. In this project, we adopt the following notion of queue-stability (same as in [11, 12]): a network is said to be *stable* if the steady-state total queue-length is finite, i.e.,

$$\limsup_{t \to \infty} \mathbb{E}\left\{ \|\mathbf{q}[t]\|_1 \right\} < \infty. \tag{2}$$

Problem formulation: Let $\bar{a}_n \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} a_n[t]$ denote the long-term average controlled arrival rate of link n. Each link n is associated with a utility function $U_n(\bar{a}_n)$, which represents the utility gained by link n when data is injected at rate $\bar{a}_n$. We assume that $U_n(\cdot)$, $\forall n$, is concave, monotonically increasing, and twice continuously differentiable. We further assume that $U_n(\cdot)$ satisfies the following *strong concavity* condition: there exist constants $0 < \phi \leq \Phi < \infty$ such that

$$\phi \leq -U_n''(a_n) \leq \Phi, \quad \forall a_n \in [0, a^{\max}], \tag{3}$$

where $a^{\max}$ is the maximum arrival rates for burst control. For example, the function $\log(\epsilon + a_n)$ with some constant $\epsilon > 0$ satisfies (3). In this project, our goal is to maximize $\sum_{n=1}^{N} U_n(\bar{a}_n)$,

subject to achievable rate region $\mathcal{C}_{\boldsymbol{\pi}[t]}$ in each time-slot and the queue-stability constraint. Putting together the models presented above yields the following joint congestion control and scheduling (JCCS) optimization problem:

JCCS:

$$\begin{aligned} &\text{Maximize} && \sum_{n=1}^{N} U_n(\bar{a}_n) \\ &\text{subject to} && \text{Queue-length stability constraint in (2)}, \\ & && s_n[t] \in \mathcal{C}_{\boldsymbol{\pi}[t]},\ a_n[t] \in [0, a^{\max}]\ \forall n, t. \end{aligned}$$

2-B The Algorithm

Our proposed heavy-ball-based network utility optimization algorithm is described in Algorithm 1:

Algorithm 1: The Heavy-Ball-Based Wireless Network Utility Optimization Algorithm.

Initialization:

1. Choose parameters $K > 0$ and $\beta \in [0, 1)$. Set $t = 0$.
2. Let all queues be empty at the initial state: $q_n[0] = 0, \forall n$.
3. Under a given K, associate each link n with a non-negative weight $w_{(K),n}$ and set $w_{(K),n}[0] = w_{(K),n}[-1] = 0, \forall n$.

Main Loop:

4. *MaxWeight Scheduler:* In time-slot $t \geq 0$, given the current weight vector value $\mathbf{w}_{(K)}[t] \triangleq [w_{(K),1}[t], \ldots, w_{(K),N}[t]]^\top$ and the current channel state $\boldsymbol{\pi}[t]$, the scheduler chooses a service rate vector $\mathbf{s}[t]$ as follows (breaking ties arbitrarily):

$$\mathbf{s}[t] = \arg \max_{\mathbf{x} \in \mathcal{C}_{\boldsymbol{\pi}[t]}} (\mathbf{w}_{(K)}[t])^\top \mathbf{x}. \tag{4}$$

5. *Congestion Controller:* For each link n, given its current weight $w_n[t]$, the data injection rate $a_n[t]$ is an integer-valued random variable that satisfies:

$$\mathbb{E}\{a_n[t] | w_{(K),n}[t]\} = \min\left\{U_n'^{-1}\left(\frac{w_{(K),n}[t]}{K}\right), a^{\max}\right\}, \tag{5}$$

$$\mathbb{E}\{a_n^2[t] | w_{(K),n}[t]\} \leq A < \infty, \quad \forall w_{(K),n}[t], \tag{6}$$

where $U_n'^{-1}(\cdot)$ represents the inverse function of the first-order derivative of $U_n(\cdot)$. In (5) and (6), $a^{\max}$ and A are some predefined positive constants.

6. *Queue-Length and Heavy-Ball Weight Updates:* Update the queue-lengths following (1). Let $\Delta q_n[t] \triangleq q_n[t+1] - q_n[t]$ be the resultant queue-length change, $\forall n$. Next, update the weights in the following (projected) ***heavy-ball*** fashion:

$$w_{(K),n}[t+1] = \left[w_{(K),n}[t] + \Delta q_n[t] + \beta(w_{(K),n}[t] - w_{(K),n}[t-1])\right]^+, \ \forall n. \tag{7}$$

Let $t = t + 1$. Go to Step 4 and repeat the scheduling and congestion control processes.

In Algorithm 1, we can see that the congestion control and scheduling components are similar to those in the QLA schemes (see, e.g., [11, 12, 27]), but with the following *key* differences: First, in both components, the weights in (4) and (5) are *not* directly using current queue-lengths (or some direct functions of current queue-lengths). It is this *separation* of weights and queue-lengths that leads to significant delay reductions. Also, we note that the weight update in (7) is motivated by the *heavy-ball* idea: It includes a β-parameterized additional *first-order memory* (or called "momentum") of the *weight change* in the previous time-slot. In contrast, the weight updates in traditional QLA algorithms are of *zero-order memory* in the sense that queue-lengths only inherit the absolute weight values in the previous time-slot. As will be seen shortly, this algorithmic structural difference necessitates new proof techniques in establishing the theoretical results.

2-C Main Theoretical Results of the Algorithm and Further Insights

The first key result in this project is on the delay reduction performance of our proposed heavy-ball algorithm:

Theorem 1 (Delay reduction and queue-stability)**.** *Under the β-parameterized heavy-ball algorithm, the scaling of the steady-state total queue-length with respect to K satisfies:*

$$\limsup_{t\to\infty} \mathbb{E}\{\|\mathbf{q}[t]\|_1\} = O\big((1-\beta)K\big) + O\big((1+\beta)\sqrt{K}\big). \tag{8}$$

Further, if β approaches 1 in such a way that $\beta = 1 - O(\frac{1}{\sqrt{K}})$, then Eq. (8) implies the following result: $\limsup_{t\to\infty} \mathbb{E}\{\|\mathbf{q}[t]\|_1\} = O(\sqrt{K})$.

Three remarks on Theorem 1 are in order: i) If β is fixed and $K \to \infty$, the first term on the right-hand-side of (8) dominates the second term and thus $\limsup_{t\to\infty} \mathbb{E}\{\|\mathbf{q}[t]\|_1\} \approx O\big((1-\beta)K\big)$. This means that a β-parameterized heavy-ball scheme leads to a delay that is approximately $(1-\beta)$*–fraction* of that of the traditional QLA methods; ii) If β is varying in relation to K, then Theorem 1 states that if $\beta \uparrow 1$ fast enough as $K \to \infty$, the total queue-length scales as $O(\sqrt{K})$, which significantly outperforms the $O(K)$ delay of the QLA algorithms. We note that this $O(\sqrt{K})$ delay is achieved *without* sacrificing any throughput and *without* requiring non-causal global statistics as in [15, 16]; iii) In some sense, including the weight changes in (7) can be viewed as a simple way of "learning" how the queues had evolved in history. Interestingly, Theorem 1 shows that even simply paying attention to "*yesterday's memory*" makes a big difference in delay performance.

Now, let $U(\mathbf{a}) = \sum_{n=1}^{N} U_n(a_n)$ be the total utility function of Problem JCCS and let $\mathbf{a}^*$ be the optimal solution. Also, let $a^{\infty}_{(K),n} \triangleq \mathbb{E}\{\min\{U_n'^{-1}(w^{\infty}_{(K),n}/K), a^{\max}\}\}$, $\forall n$, be the mean steady-state congestion control rates offered by our heavy-ball algorithm (the existence of steady-state will be proved in the appendices). Further, we let $\mathbf{a}^{\infty}_{(K)} \triangleq [a^{\infty}_{(K),1}, \ldots, a^{\infty}_{(K),N}]^{\top}$. The next result states that our proposed heavy-ball algorithm is *utility-optimal* and the optimality is independent of β:

Theorem 2 (Utility-optimality)**.** *Under Algorithm 1 and for some given K, the mean of the stationary rate vector $\mathbf{a}^{\infty}_{(K)}$ satisfies $\|\mathbf{a}^{\infty}_{(K)} - \mathbf{a}^*\| = O(1/\sqrt{K})$. Also, the optimal utility objective value can be bounded as $U\big(\mathbf{a}^{\infty}_{(K)}\big) \geq U(\mathbf{a}^*) - O(1/K)$. Hence, $\mathbf{a}^{\infty}_{(K)}$ converges to $\mathbf{a}^*$ asymptotically as K increases.*

Our third result is on the convergence speed performance. In this project, the notion of convergence speed is defined in terms of the fewest number of time-slots that the sequence $\{\mathbf{w}_{(K)}[t]\}$ takes so that the resultant sequence $\{\mathbb{E}\{\mathbf{a}_{(K)}[t]|\mathbf{w}_{(K)}[t]\}\}$ reaches the $O(\frac{1}{\sqrt{K}})$-neighborhood of $\mathbf{a}^*$ stated in Theorem 2.

Theorem 3 (Linear convergence rate)**.** *Let K and β be chosen as $K \in (\frac{\Phi}{4}, \infty]$ and $\beta \in \big[\max\big\{0, \frac{\Phi}{2K} - 1\big\}, 1\big)$. Then, $\{\mathbb{E}\{\mathbf{a}_{(K)}[t]|\mathbf{w}_{(K)}[t]\}\}$ converges linearly[2] with a factor $R_{(K,\beta)} \leq \max\big\{\sqrt{\beta}, |1+\beta-\phi/K|-\sqrt{\beta}, |1+\beta-\Phi/K|-\sqrt{\beta}\big\}$. Moreover, minimizing the upper-bound of $R_{(K,\beta)}$ yields: $R^* = (\sqrt{\Phi}-\sqrt{\phi})/(\sqrt{\Phi}+\sqrt{\phi})$, which is obtained by $K^* = (\sqrt{\Phi}+\sqrt{\phi})^2/4$ and $\beta^* = (\sqrt{\Phi}-\sqrt{\phi})^2/(\sqrt{\Phi}+\sqrt{\phi})^2$.*

Theorem 3 says that we can optimize K and β to achieve $R^* = (\sqrt{\kappa}-1)/(\sqrt{\kappa}+1)$, where $\kappa \triangleq \Phi/\phi$ is the condition number [21]. The optimized R^* is always *smaller* compared to that of the QLA approaches, where $R_{\rm QLA} = (\kappa-1)/(\kappa+1)$ (cf. e.g., [11]), thus implying a faster convergence. Moreover, this convergence speedup phenomenon is even more pronounced when κ is large (i.e., the problem is ill-conditioned).

Finally, combining the results in Theorems 1–3, we obtain the following *three-way trade-off* between utility, delay, and convergence speed under the proposed heavy-ball algorithm:

Theorem 4 (Three-way performance trade-off)**.** *Given control parameters $K \in (\frac{\Phi}{4}, \infty]$ and $\beta \in \big[\max\big\{0, \frac{\Phi}{2K}-1\big\}, 1\big)$, the proposed heavy-ball algorithm achieves an $O(1/K)$ in utility-optimality gap, $O\big((1-\beta)K\big) + O\big((1+\beta)\sqrt{K}\big)$ in delay, and $O[\log(\sqrt{K})(-\log^{-1}(1+\beta-\sqrt{\beta}))]$ in convergence time.*

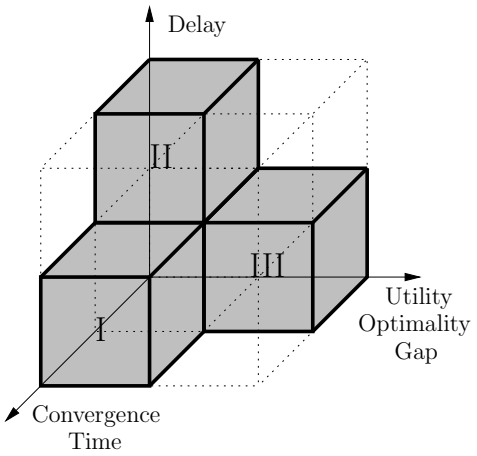
The proofs of Theorems 1–4 will be provided in the appendices. In what follows, we will further discuss the key insights of the theoretical results in Theorems 1–4. There several key insights implied by the aforementioned theoretical results.

1) Thee-way performance trade-off relationship: Collectively, Theorems 1–4 suggest a *new* three-way trade-off relationship where, by appropriately selecting K and β, one can *simultaneously improve two out of the three performance metrics by trading-off the third*. To facilitate better understanding, we illustrate this three-way trade-off relationship in Fig. 3. In Fig. 3, the arrow of each axis is pointing toward worse performance in utility, delay, and convergence, respectively. The regions I, II, and III represent three types of trade-off relationships achieved under our heavy-ball algorithm, and the table in Fig. 3 illustrates how each region corresponds to the settings of the two control knobs K and β.

First, Region I in Fig. 3 represents "*achieving both utility-optimality and low-delay by setting a large K and choosing β close to 1, at the cost of slower convergence*." To see this, we first note from Theorem 2 that a large K implies small utility-optimality gap $O(1/K)$. Also, by choosing β close to 1, Theorem 1 implies that the $(1-\beta)$–fraction delay reduction is significant. However, when $K \to \infty$ and $\beta \to 1$, it is not difficult to verify from Theorem 3 that:

$$\lim_{\substack{\beta\to 1\\ K\to\infty}} R_{(K,\beta)} \leq \lim_{\substack{\beta\to 1\\ K\to\infty}} \Big\{\max\big\{\sqrt{\beta}, |1+\beta-l/K|-\sqrt{\beta}, \\ |1+\beta-L/K|-\sqrt{\beta}\big\}\Big\} \to 1.$$

[2]We say that a sequence $\{x_k\}_{k=1}^{\infty}$ converges linearly to x^* if there exists a factor $R \in (0,1)$ such that $\|x_{k+1}-x^*\| \leq R\|x_k - x^*\|$ for all k.



	K	β
I	Large	Close to 1
II	Large	Optimized for Rate of Convergence
III	Optimized for Rate of Convergence	Close to 1

Figure 3: An illustration of the three-way trade-off relationship.

That is, as $K \to \infty$ and $\beta \to 1$, the worst case convergence rate factor $R_{(K,\beta)}$ asymptotically approaches 1, which implies an increasingly slower convergence speed.

Second, Region II represents "*achieving utility-optimality and fast-convergence by setting a large K and optimizing β, at the cost of less delay performance gain.*" To see this, we again note from Theorem 2 that a large K implies small utility-optimality gap $O(1/K)$. Also, by Theorem 3, we can optimize β under a given K to minimize the convergence factor $R_{(K,\beta)}$ to increase the convergence speed. However, the obtained β is not necessarily close to 1 and thus the delay performance gain may not be dramatic. We note that, in Region II, even though the optimized β may not entail dramatic delay reduction, one still enjoys the benefit of $(1-\beta)$–fraction delay compared to the QLA approaches, according to Theorem 1.

Lastly, Region III represents "*achieving low-delay and fast convergence by setting β close to 1 and optimizing K, at the cost of larger utility-optimality gap.*" To see this, we note from Theorem 1 that we can first push β close to 1 to achieve low delay. With the given β, by Theorem 3, we can optimize K to minimize the convergence factor $R_{(K,\beta)}$ to increase the convergence speed. However, the obtained K is not necessarily large and thus the utility-optimality gap may not be small.

2) Intuition behind large delay reduction: Before rigorously proving the $(1-\beta)$–fraction delay reduction result, we first give some high-level intuitions to shed light on why the proposed heavy-ball method enjoys a large delay reduction over the back-pressure-based algorithms. To this end, we take a closer look at the back-pressure-based algoritihms. From the fundamental connection between the back-pressure-based algorithms and the Lagrangian dual decomposition framework (see, e.g., [11–13, 28]), one can see that the back-pressure-based algorithms can be interpreted as using queue-lengths as the Lagrangian dual variables to solve the deterministic Problem D-JSSC. A disadvantage of such an approach is that a large amount of packets are kept in the queues *only* for the purpose of maintaining the "right pressure" at the level of $\mathbf{w}^*_{(K)}$. It is important to recognize that this "queue-lengths as dual" approach is *not* necessary since the optimal dual solution $\mathbf{w}^*_{(K)}$ is merely a mathematical construct and needs not be tied with any physical quantity, such as queue-lengths. In theory, one has the freedom to use any quantity to represent the dual variables $\mathbf{w}$.

Indeed, several delay reduction methods based on the above observation have been proposed

in the literature to reduce queueing backlogs. The idea that is in closest spirit to our heavy-ball approach can be found [16], where the authors proposed to place $\mathcal{W}_n \in \left[0, w^*_{(K),n}\right)$ "place-holder bits" in each queueing buffer n and use the weight $w_n[t] \approx q_n[t] + \mathcal{W}_n$ to conduct scheduling.[3] However, it turns out that finding an appropriate value for each $\mathcal{W}_n$ is a tricky task since doing so requires the knowledge of $\mathbf{w}^*_{(K)}$, which is usually unavailable at the initial state of the algorithm. The solve this problem, a per-iteration learning step was proposed in [17] to learn $\mathbf{w}^*_{(K)}$ in an online fashion. However, adding a learning component in each iteration significantly increases the overall complexity of the algorithm. What is worse is that these "place-holder-bits" approaches in [16, 17] result in non-zero packet dropping probability, which could be highly undesirable in practice.

Now, consider the heavy-ball weight update step in (7), which is re-stated as follows (omitting the index "(K)" for brevity):

$$w_n[t+1] = [w_n[t] + \Delta q_n[t] + \beta(w_n[t] - w_n[t-1])]^+, \quad \forall n.$$

It can be seen that the momentum term $\beta(w_n[t] - w_n[t-1])$ plays a similar role as the place-holder bits in the sense that $\beta(w_n[t] - w_n[t-1])$ effectively *reduces* the size of required $\Delta q_n[t]$ to maintain the right level of $w_n[t]$. However, the use of momentum term has the following advantages over the place-holder bits: i) Unlike the artificial notion of place-holder bits that is difficult to set appropriately, the momentum term corresponds to two time-slots of weight memory, which not only possesses a nice physical meaning but also lends itself to simple implementation; ii) The β-parameterized momentum term can also be viewed as a simple exponential moving average scheme to implicitly learn and adapt to the unknown $\mathbf{w}^*_{(K)}$, thus eliminating the need for an explicit per-iteration learning step as in [17]; and (iii) Rather than having the queues head-start at an overly aggressive $\mathcal{W}_n$-value and resulting in packet dropping, the heavy-ball weight update scheme evolves gracefully and does *not* incur any packet dropping thanks to the implicit adaptive learning, and thus retaining full throughput-optimality.

3) Why do we need both K and the β-parameterized momentum? Given the above interpretation that the β-parameterized momentum term $\beta(w_n[t] - w_n[t-1])$ effectively reduces the queue-sizes to $(1-\beta)$–fraction, one may be tempted to ask the following legitimate question: *Could we directly use $\frac{1}{1-\beta}\mathbf{q}[t]$ as weights in back-pressure to perform max-weight scheduling in (4) in order to achieve the same $[O(1/K), O((1-\beta)K)]$ utility-delay trade-off?* Unfortunately, the answer to this question is "No." If one uses $\frac{1}{1-\beta}\mathbf{q}[t]$ to perform scheduling, the delay does reduce to $O((1-\beta)K)$. However, in this case, the congestion control decision in (5) effectively becomes $\min\left\{U_n'^{-1}\left(\frac{q_n[t]}{(1-\beta)K}\right), a^{\max}\right\}$, which is equivalent to reducing K to $(1-\beta)K$ in congestion control. Hence, according to Theorem 2, the utility-optimality gap would grow to $O(\frac{1}{(1-\beta)K})$. The analysis above reveals the key benefit and fundamental difference of using the β-parametrized momentum: Without using the momentum term, it is not possible to separate the coupling between scheduling and congestion control. In contrast, the use of the momentum term provides *one more degree of freedom* in separating the controls between utility-optimality and delay, thus breaking the gridlock between utility-optimality and delay.

[3] We use "$\approx$" here because, in [16], some additional projections are needed for to maintain non-negativity of $q_n[t]$ and $w_{[n]}[t]$.

2-D EMANE Simulations

Based on the key theoretical results discussed in previous sections, a **key focus** of this project is to test and verify these theoretical results on the so-called EMANE simulation platform.

1) The EMANE Simulator: The Extendable Mobile Ad-hoc Network Emulator (EMANE) is an advanced framework for real-time modeling of mobile network systems. The EMANE components focus on real-time modeling of link and physical layer connectivity so that network protocol and application software can be experimentally subjected to the same conditions that are expected to occur in real-world mobile, wireless network systems. EMANE architecture provides for Network Emulation Modules (NEMs) that can be associated with computer system (real or virtualized) network stacks as interfaces. The EMANE framework further provides an event-driven control bus and logging facilities.

EMANE is also a modular framework and multiple, different NEMs can be developed, instantiated, and associated with emulated mobile network nodes. Additionally, many other components of EMANE are "pluggable." For example, the interface mechanism between NEMs and emulated system network stacks is also a component that can be refactored and replaced (on a per-NEM basis) via run-time configuration. Similarly, other aspects of EMANE, such as the control bus (default is IP multicast over a LAN (e.g. Gbps Ethernet), logging services, etc., are distinct components that can be extended or replaced. And while Internet Protocol (IP) is directly supported by EMANE, it is also possible to use the EMANE framework and its NEMs with custom, non-IP protocol stacks. Due to the above salient features of EMANE, we adopt EMANE as our primary simulation platform to test and verify our proposed algorithms.

2) Challenges of EMANE Simulations: Although the EMANE simulation software is a powerful wireless network simulation platform, there remain several major technical challenges to implement, test, and verify the performance of our proposed HeavyBall algorithm on EMANE:

- Our HeavyBall algorithm needs a series of cross-layer functionalities, which are not in readily available in EMANE. Specifically, we need i) SNR information from the PHY layer in the MAC layer scheduling design; ii) queue-length information in the congestion control at the transport layer from the MAC layer.

- We need specialized transport layer to generate packets in a dynamic fashion in the congestion controller component of our HeavyBall algorithm. For example, we can use the the MGEN traffic generator. However, it is difficult to integrate MGEN with EMANE and generate packets with a dynamically adjustable source injection rate.

3) Our Solutions: To address the above challenges in EMANE simulation implementation, our first approach is to develop three components in EMANE: i) Pathloss Holder; ii) Scheduler Module; and iii) Dynamic MGEN (Congestion Control). The overall architecture of our developed components is shown in Fig. 4. Since we modified the internal components in EMANE, we call this approach the "Customized Approach."

3-1) The "Customized" Approach: Specifically, in Fig. 4, the purpose of the pathloss holder is to obtain the SNR of each node based on the pathloss information in the physical layer. In the scheduler module, based on the extracted SNR from the physical layer, we can emulate channel state changes, including bandwidth and fading. Then, based on SNR and queueing information, we are able to implement our MaxWeight-type scheduling algorithm. The dynamic MGEN component

is at the top of Fig. 4. In what follows, we describe the challenges in developing each component in greater detail and how we addressed these challenges:

1) Pathloss Holder: The main difficulty in implementing the pathloss holder component is that, in EMANE, the MAC and PHY layers are decoupled: only up/down stream data packets can be exchanged in the original EMANE. To solve this problem, we created the Pathloss Holder component for the MAC layer to get access to SNR information when the PHY layer updates the channel state information (CSI).

2) Scheduler Module: To implement the MaxWeight-type scheduler, we modified the TDMA scheduler in EMANE so that scheduling algorithms (not limited to our HeavyBall) that need queue-length, CSI, and other information can be implemented.

3) Dynamic MGEN: The main challenge in implementing dynamic congestion lies in how to implement the dynamic congestion control that can react to queue-lengths and changing bandwidths. In our software development, we have attempted three different approaches. The first two failed, and the third succeeded. Our first approach uses multi-thread programming in EMANE to implement our congestion control function. However, due to the complex multi-thread mechanism in EMANE, this approach resulted in many OS crashes that we did not know how to fix. In our second approach, we tried to CORE firewall (CORE is another GUI tool that works with EMANE) to adjust the congestion control rate. However, what we found is that CORE firewall cannot achieve source rate control that is accurate enough to meet our purposes. Lastly, we decided to create a wrapper on top of the MGEN component to achieve an accurate and dynamically adjustable source injection rate. We called our congestion control component Dynamic MGEN. This component contains a "data tube" (socket) to receive (virtual) queue-lengths/weights in each time-slot. In Section 3, we will show some experimental results to show that this "customized" approach works very well and indeed verifies all our theoretical predictions for the HeavyBall algorithm.

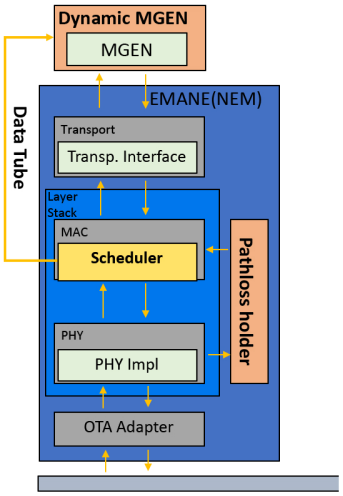


Figure 4: System architecture of our "customized" approach in EMANE to implement HeavyBall.

3-2) The Shim Layer Approach: Although the "customized" approach works well in practice, upon communicating with AFRL, we learned this is not a preferred way to implement our EMANE simulations. Specifically, the first problem with the "customized" approach is that there are significant modifications at the MAC layer, which include modifications to the MAC layer base model and modifications to the queue manager in EMANE. These could lead to potential compatibility risks with other PHY layer implementation in EMANE. The second problem with the "customized" approach is that there are modifications to the PHY layer to handle the new output (e.g., CSI). This could lead to potential compatibility risks with future MAC layer implementations in EMANE. To address this problem, we proposed to re-implement our scheduling component using the so-called "shim layer" approach in EMANE.

Simply speaking, in EMANE, a shim layer is a special library that can transparently intercepts API calls, changes the arguments passed, and/or handles operations themselves or redirects those operations elsewhere. Applications of a shim layer include to support new API in old environment or to support old API in new environments. The reason we choose the shim layer approach in

EMANE implementation is that we will no longer need to touch EMANE's MAC and PHY layers. As shown in Fig. 5, the location of our developed shim layer in the overall architecture is sitting between the MAC and the transport layers. The functionalities of the scheduling shim layer component is shown in Fig. 6. Our shim layer module contains the following functionalities:

1) Initialization: This function initializes the HeavyBall shim layer.

2) Queue Manager: This function i) generates/maintains queues; ii) exchanges queue-length information with the scheduler; and iii) exchanges queue-length information with the congestion controller.

3) HeavyBall (Momentum) Weight Updating: This function is responsible for all HeavyBall weight updates in the scheduling module.

4) MaxWeight Scheduler: This function performs the MaxWeight scheduling in the shim layer. Note that, under single-hop networks with only one link being activated in each time-slot, the MaxWeight scheduler is simplified to a greedy algorithm that always selects the link that has the largest queueing backlogs.

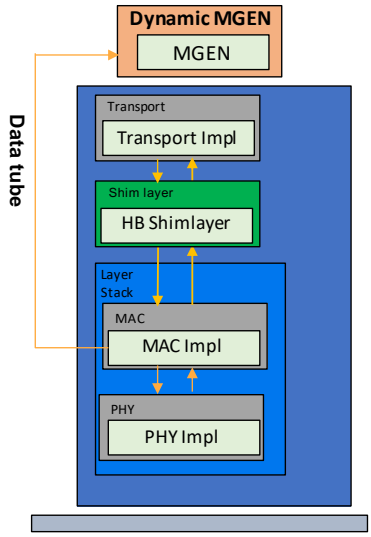


Figure 5: System architecture of our "shim layer" approach in EMANE.

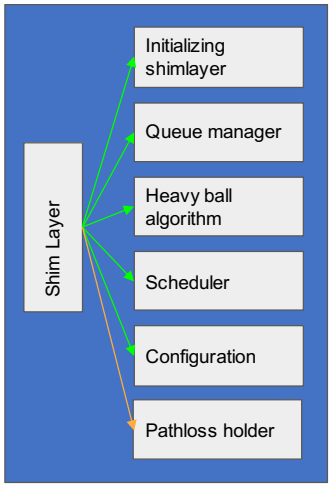


Figure 6: The zoom-in view of our proposed shim layer architecture.

5) CSI Collection: This function collects all CSI from the PHY layer, i.e., playing the same role as the pathloss holder in the "customized" approach.

6) Time-Slotted Radio Module Detection: This function will detect the underlying radio module in use. It accepts time-slotted radio modules (e.g., TDMA, Link 16) in EMANE. It will reject other non-time-slotted radio modules (e.g., CSMA). As a result, our shim layer module is robust in terms of radio module compatibility.

3-3 Integration with CORE: To fully integrate with the EMANE/CORE ecosystem for wireless network simulations, in this project, we will integrate the control/configuration interface of our develop HeavyBall shim layer module in the CORE software package. CORE (Common Open Research Emulator) is a tool for building virtual networks. As an emulator, CORE builds a representation of a real computer network that runs in real time, as opposed to simulation, where abstract models are used. The live-running emulation can be connected to physical networks and routers. It provides an environment for running real applications and protocols, taking advantage of tools provided by the Linux operating system. CORE is typically used for network and protocol research, demonstrations, application and platform testing, evaluating networking scenarios, security studies, and increasing the size of physical test networks. The CORE source consists of several different programming languages for historical reasons. Current development focuses on the Python modules and daemon.

During the development of our "customized" approach, we developed a simple Python-based graphical user interface (GUI) for demonstrating our HeavyBall algorithm. In the final stage of

this project, we will integrate the shim layer into CORE so that the shim layer can be controlled and configured more easily. This integration also directly contributes to the development of the EMANE/CORE ecosystem for *cross-layer* wireless network control and optimization.

3 Results and Discussions

In this section, we conduct simulation numerical studies to verify the theoretical results presented in Section 2. Although our focus in this project is to develop an EMANE-based simulation platform to test and verify our HeavyBall algorithm, it is insightful to present simple MATLAB-based simulations as a comparison benchmark. Thus, we will illustrate our MATLAB-based simulation results in Section 3-A. Then, we will illustrate and discuss our EMANE-based simulation results in Section 3-B.

3-A MATLAB-Based Simulations

To clearly visualize the key insights of our theoretical results and not being blurred by random noises, we first use a three-link non-fading cellular network as an example. In this network, we assume that each link has unit capacity. Due to interference, only one link can be activated in each time-slot. In this example, we use $\log(0.001 + a)$ as the utility function for each link, i.e., the well-known proportional fairness metric [14]. Due to the symmetry of the network structure, it is easy to see that the optimal congestion control rates are $\bar{a}_1^* = \bar{a}_2^* = \bar{a}_3^* = \frac{1}{3}$.

To see the impact of β on delay and convergence speed, we fix $K = 25$ and increase β from 0 to 0.99 (note that $\beta = 0$ corresponds to the QLA approach). Because of the symmetry of the setting, we only plot the results of link 1. As shown in Fig. 7, as β increases, the average queue-lengths are 74.6, 37.4, 14.8, and 1.1, respectively, which corroborates the $(1-\beta)$–fraction reduction result in Theorem 1. We can see from Fig. 8 that, for all choices of β, the congestion control rates all converge to the optimal solution, confirming Theorem 2 that utility-optimality is independent of β. However, changing β has a significant impact on the convergence speed. In Fig. 8, as β increases from 0 to 0.99, the convergence speed initially increases, peaks at $\beta = 0.8$, and then decreases. Interestingly, we note from Fig. 7 and Fig. 8 that, by setting $\beta = 0.99$, both utility-optimality and low-delay can be achieved at the cost of slower convergence speed, hence confirming Theorem 4.

Next, we increase K from 25 to 100 and conduct another set of experiments on the same network. The results are shown in Fig. 9 and Fig. 10, respectively. With a larger K, the congestion control rates again converge to the same optimal solution with a smaller variance, but at the cost of larger delay and slower convergence. This again confirms the results in Theorems 1–4.

Now, we study the convergence of the congestion control rates and steady-state queue-lengths in a larger 15-user cellular downlink system with fading environments. Again, we assume that only one user can be activated in each time-slot. In each time-slot, the supportable rate of each user under fading is a Gaussian random variable with mean 5 and standard deviation 3. First, we fix $K = 100$ and set $beta$ from 0 (i.e., back-pressure), 0.35, 0.65, 0.95, respectively. Due to the symmetry of the system, we only plot the congestion control rate and queue-length of user 1 in Fig. 11 and Fig. 12, respectively. We can see from Fig. 11, as β increases, the queue-length also monotonically decreases and the queue-length reduction again follows the $(1-\beta)$-fraction reduction law stated in Theorem 1. Meanwhile, from Fig. 12, we can see that the congestion

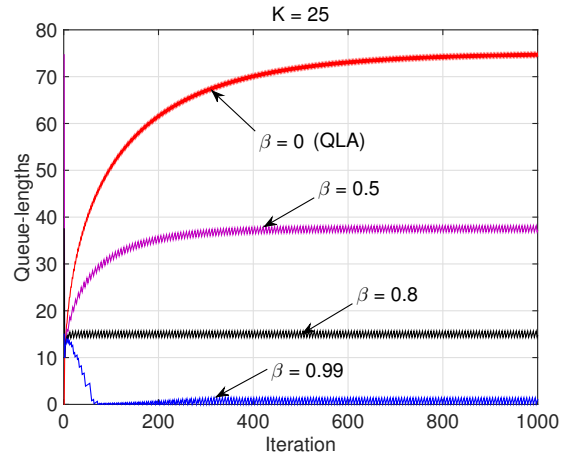


Figure 7: The impact of β on queueing delay.

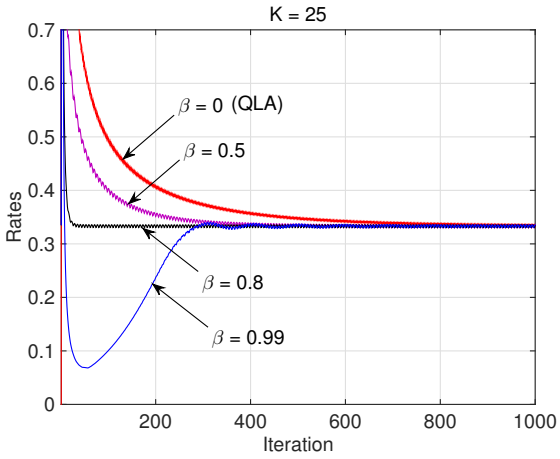


Figure 8: The impact of β on convergence speed.

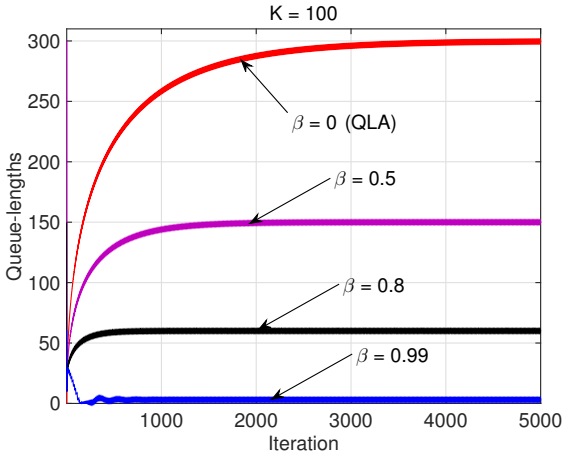


Figure 9: The impact of K on queueing delay.

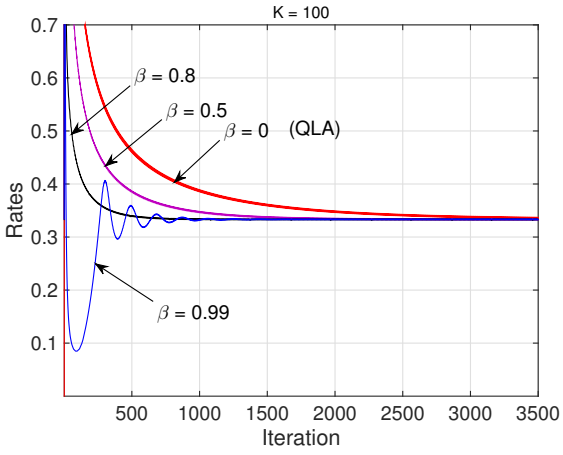


Figure 10: The impact of K on convergence speed.

control rates under different β all converge to the same optimal solution, which approximately $\frac{1}{3}$. Also, the convergence time initially shortens as β increases. However, when β is large ($\beta = 0.95$), the convergence becomes slower and exhibits larger variance. This shows that the same three-way trade-off effect continues to hold in this larger system with fading, i.e., we can achieve utility-optimality and low-delay by choosing a large β, but at the cost of slower convergence.

Next, we increases K from 100 to 300 and conduct another set of experiments to observe its effect. The results are illustrated in Fig. 13 and Fig. 14, respectively. We can see that, with a larger K, the congestion control rates again converge to the same optimal solution with a smaller variance, but at the cost of larger delay and longer convergence time. This again confirms our theoretical result in Theorems 1 and 2, and 4.

Lastly, we compare the delay scaling with respect to K under QLA and our heavy-ball algorithm, respectively. Here, as K increases, we let $\beta \uparrow 1$ as $\beta = 1 - \frac{1}{\sqrt{K}}$. As expected, in Fig. 15, the total queue-length of QLA exhibits the well-known $O(K)$ linear scaling law and is significantly

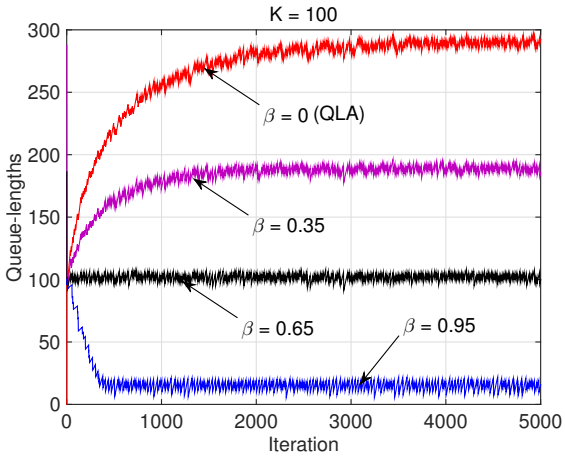


Figure 11: The impact of β on queueing delay for a 15-user cellular downlink with fading ($K = 100$).

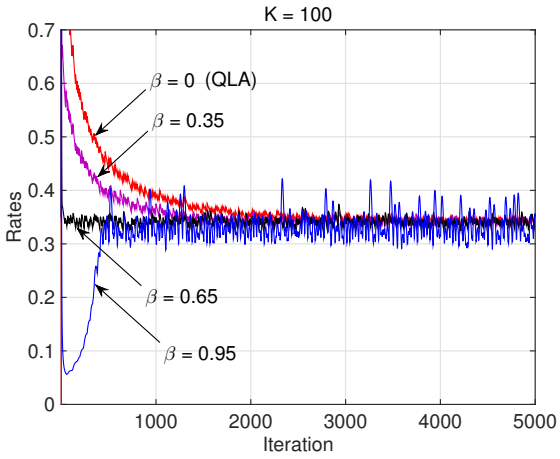


Figure 12: The impact of β on convergence speed for a 15-user cellular downlink with fading ($K = 100$).

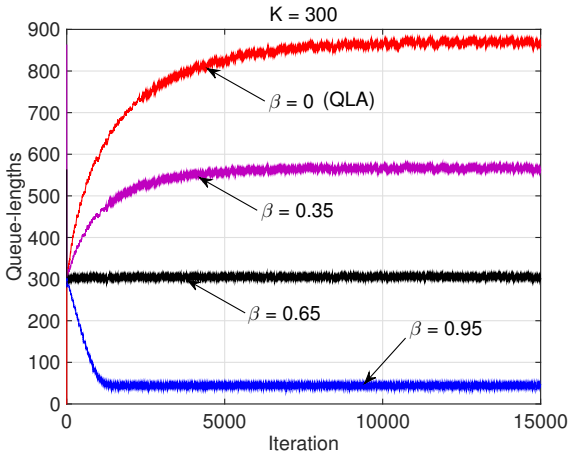


Figure 13: The impact of β on queueing delay for a 15-user cellular downlink with fading ($K = 300$).

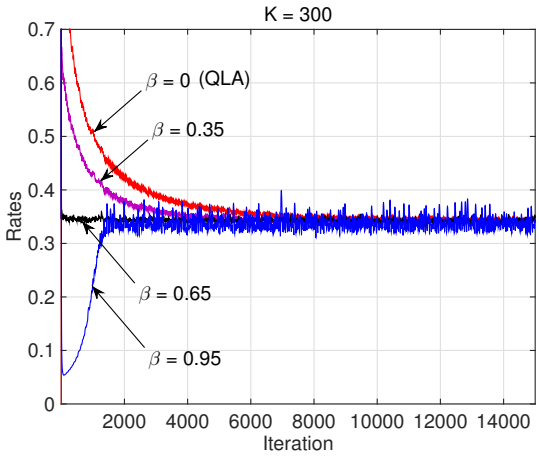


Figure 14: The impact of β on convergence speed for a 15-user cellular downlink with fading ($K = 300$).

larger than that of our heavy-ball algorithm. Further, in the zoom-in view of the heavy-ball results in Fig. 16, we can see that the total queue-length increases as $4.5\sqrt{K}$, which perfectly matches the $O(\sqrt{K})$ theoretical result in Theorem 1.

3-B EMANE-Based Simulations

As discussed in Section 2-D, a key focus on this project is to implement our HeavyBall algorithm in the EMANE/CORE simulation platform. Here, we tested HeavyBall in a three-node wireless network with momentum coefficient $\beta \in \{0.3, 0.6, 0.9\}$. The simulation outcome of EMANE is shown in Fig. 17. We can see that the EMANE simulation results follow exactly the same trends as in our theoretical simulations in MATLAB. That is, the congestion control rates all converge to the

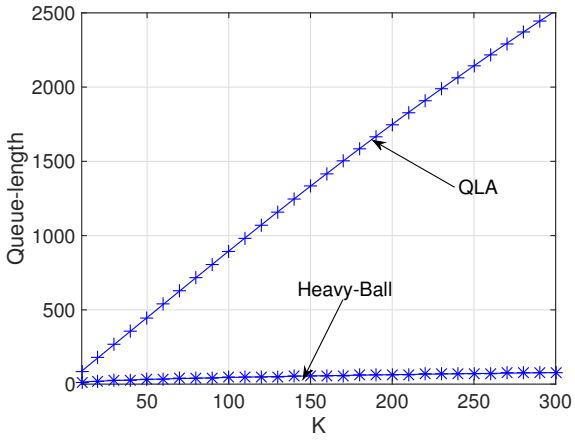


Figure 15: The linear scaling law with respect to K under the back-pressure algorithm.

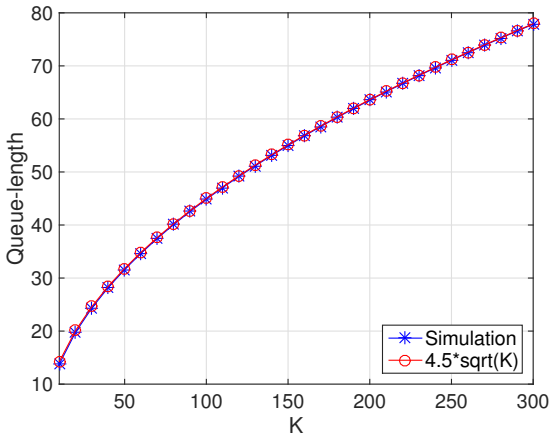


Figure 16: The $O(\sqrt{K})$ scaling law under our heavy-ball algorithm with β approaches 1 at rate $1-\beta = O(\frac{1}{\sqrt{K}})$.

same optimal solution regardless of the choices of the β-value. Meanwhile, as β approaches 1, we achieve lower and lower delay (reflected in the total queue-lengths). However, this performance gain in delay is achieved at the expense of slower convergence performance (tend to be unstable as β becomes very close to 1).

As mentioned in Section 2-D, upon successfully implementing the HeavyBall shim layer module, we will integrate the shim layer with the CORE software package to provide a better user interface. Due to the insufficient amount of time at the end of this project, this integration was not fully completed yet. However, we are still continuing this effort despite the project had ended in January 2021. The latest progress of this integration is shown in Figs. 18 and 19

We can see from Fig. 18 that the HeavyBall menu option has been added to the WLAN component in CORE and can be enabled to configure the options of the HeavyBall shim layer module. By selecting the "heavyball" radio button and clicking the "heavyball options" button, we jump to the pop-up window as shown in Fig. 19. In Fig. 19, we can see that in addition to all the parameter settings that are inherited from the original TDMA module, we have added two additional input entries. The "HeavyBall Beta value should be between 0 and 1" field allows you to enter a value for the momentum coefficient β (the second option from the bottom of the window). Moreover, we also added another option that will turn on the live graph plotting for HeavyBall as demonstrated as shown in Fig. 17.

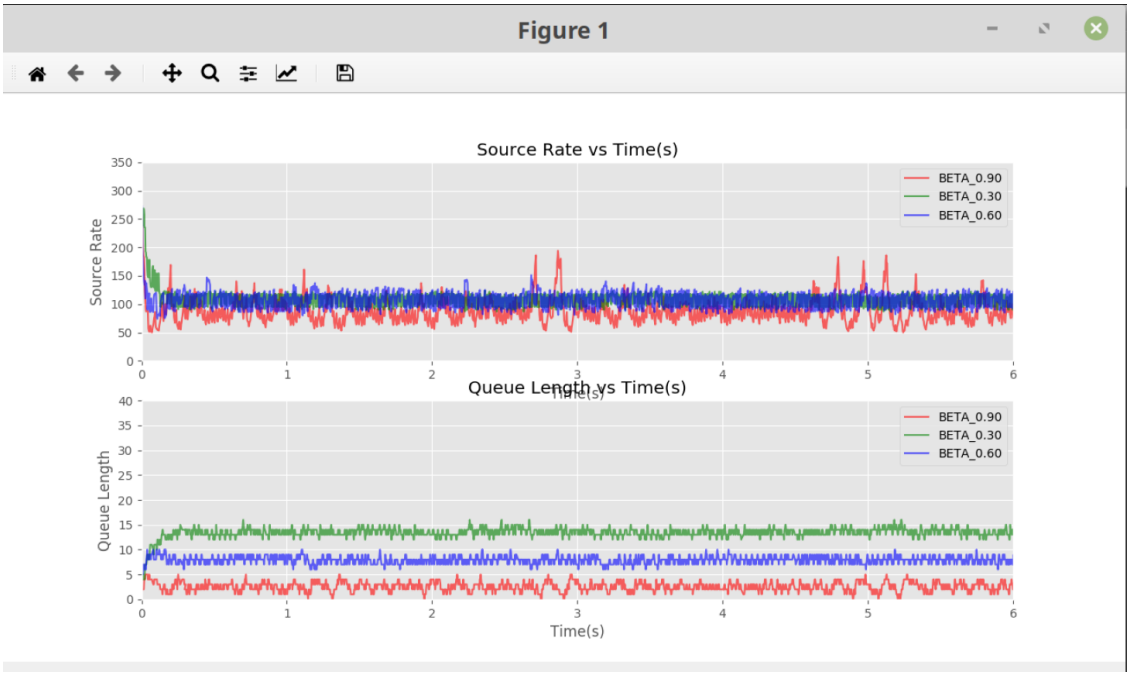


Figure 17: An illustration of simulation output of EMANE.

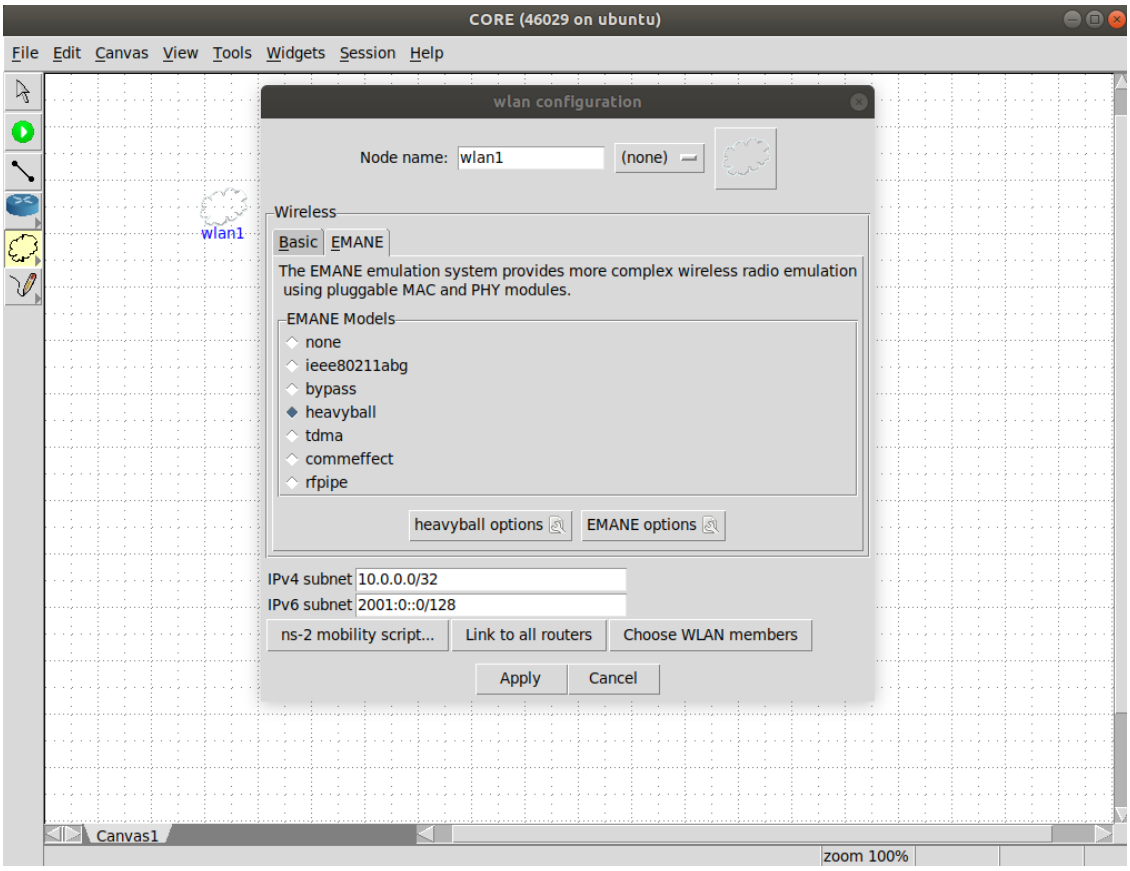


Figure 18: A screenshot showing the integration of our HeavyBall shim layer component in the CORE software package.

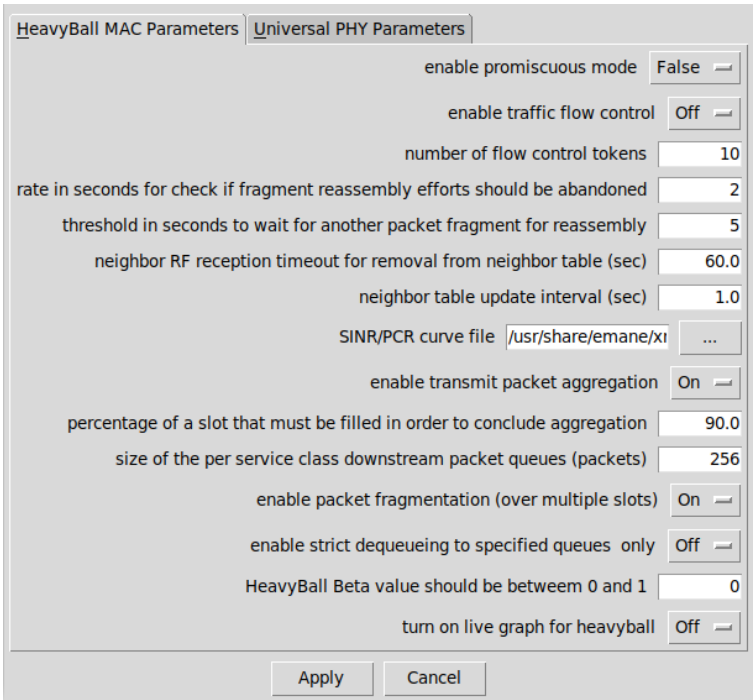


Figure 19: A screenshot showing the integration of our HeavyBall shim layer component in the CORE software package.

4 Conclusions

In this paper, we have developed a new heavy-ball algorithmic framework for network utility optimization in wireless networks. Compared to the traditional queue-length-based algorithms, our proposed heavy-ball algorithmic framework offers not only utility-optimality and queue-stability, but also fast-convergence and low-delay. Our main contributions in this paper are three-fold: i) We have proposed a heavy-ball joint congestion control and scheduling/routing framework that is well-suited for implementation in practice; ii) we have rigorously shown the utility-optimality of the proposed heavy-ball algorithmic framework and characterized the delay reduction and convergence speed performances; and iii) we offered design rules for optimal selection of systems parameters, as well as insights on an elegant three-way trade-off between utility, delay, and convergence speed. Collectively, these results serve as an exciting first step toward a cross-layer network control and optimization theory that leverages "momentum/memory" information. Heavy-ball-based cross-layer network optimization is an important and yet under-explored area. Future research topics may include, e.g., heavy-traffic delay performance analysis for heavy-ball-based scheduling algorithms, time-varying adaptive memory weight adjustments, and investigating the impact of higher order memory on network utility, delay, and convergence performances.

References

[1] L. Huang, J. Walrand, and K. Ramchandran, "Optimal smart gid tariff," in *Proc. Information Theory and Applications Workshop*, San Diego, CA, February 2012.

[2] M. J. Neely, A. S. Tehrani, and A. G. Dimakis, "Efficient algorithms for renewable energy allocation to delay tolerant consumers," in *Proc. IEEE SmartGridComm*, October 2010.

[3] M. J. Neely, R. Urgaonkar, B. Urgaonkar, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *Proc. ACM Sigmetrics*, June 2011.

[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vols. I and II*. Boston, MA: Athena Scientific, 2005 and 2007.

[5] M. J. Neely and L. Huang, "Dynamic product assembly and inventory control for maximum profit," in *Proc. IEEE CDC*, Atlanta, GA, December 2010.

[6] J. G. Dai and W. Lin, "Maximum pressure polices in stochastic processing networks," *Operations Research*, vol. 53, no. 2, March-April 2005.

[7] L. Jiang and J. Walrand, "Stable and utility-maximizing scheduling for stochastic proccessing networks," in *Proc. Allerton Conference on Communication, Control, and Computing*, 2009.

[8] P. Varaiya, *The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections*, ser. Advances in Dynamic Network Modeling in Complex Transportation Systems. New York, NY: Springer, 2013.

[9] T. Le, P. Kovacs, N. Walton, H. L. Vu, L. Andrew, and S. Hoogendoorn, "Decentralized signal control for urban road networks," ArXiv Technical Report, arXiv:1310:0491v1, Tech. Rep., 2013.

[10] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[11] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, Aug. 2006.

[12] M. J. Neely, E. Modiano, and C.-P. Li, "Faireness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[13] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.

[14] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[15] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1489–1501, Aug. 2006.

[16] L. Huang and M. J. Neely, "Delay reduction via lagrange multipliers in stochastic network optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 4, pp. 842–857, Apr. 2011.

[17] L. Huang, X. Liu, and X. Hao, "The power of online learning in stochastic network optimization," in *Proc. ACM Sigmetrics*, Austin, TX, Jun.16-20, 2014, pp. 153–165.

[18] J. Liu, C. H. Xia, N. B. Shroff, and H. D. Sherali, "Distributed cross-layer optimization in wireless networks: A second-order approach," in *Proc. IEEE INFOCOM (***Best Paper Runner-up Award***)*, Turin, Italy, Apr. 14-19, 2013.

[19] J. Liu, N. B. Shroff, C. H. Xia, and H. D. Sherali, "Joint congestion control and routing optimization: An efficient second-order distributed approach," *IEEE/ACM Trans. Netw.*, 2015, accepted, to appear.

[20] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[21] ——, *Introduction to Optimization*. New York, NY: Optimization Software, Inc., May 1987.

[22] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999.

[23] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001, pp. 123–134.

[24] A. Laksmikantha, C. Beck, and R. Srikant, "Robustness of real and virtual queue-based active queue management schemes," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 81–93, Feb. 2005.

[25] P. Ochs, T. Brox, and T. Pock, "iPiasco: Inertial proximal algorithm for strongly convex optimization," *Journal of Mathematical Imaging and Vision (JMIV)*, 2015.

[26] E. Ghadimi, I. Shames, and M. Johansson, "Multi-step gradient methods for networked optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 21, pp. 5417–5429, Nov. 2013.

[27] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.

[28] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE CDC*, Atlantis, Paradise Island, Bahamas, Dec. 2006, pp. 1484–1489.

[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York, NY: Cambridge University Press, 1990.

[30] J. Liu, A. Eryilmaz, N. B. Shroff, and E. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," *Technical Report*, Jul. 2015. [Online]. Available: https://www.dropbox.com/s/gyptfed3lssd7r2/HeavyBall_JCCR_TR.pdf?dl=0

A Proofs of the Main Theorems

In this subsection, we provide (sketched) proofs for the theorems in Section 2-C. We relegate some detailed proof derivations to appendices for better readability. We begin by defining a special block-structured matrix $\mathbf{\Gamma} \in \mathbb{R}^{2N\times 2N}$ as follows:

$$\mathbf{\Gamma} \triangleq \begin{bmatrix} (1+\beta)\mathbf{I}_N & -\beta\mathbf{I}_N \\ \mathbf{I}_N & \mathbf{0}_N \end{bmatrix}, \tag{9}$$

where $\beta \in [0,1)$ is the same momentum parameter used in our heavy-ball algorithm. Next, we prove a key lemma about the eigenvalues of $\mathbf{\Gamma}$ that will be useful in establishing Theorem 1.

Lemma 1 (Eigen-spectrum of $\mathbf{\Gamma}$)**.** *$\mathbf{\Gamma}$ only has two distinct eigenvalues: β and 1, and both eigenvalues are of algebraic multiplicity N. Hence, $\mathbf{\Gamma}$ is a non-expansive linear transformation in $\mathbb{R}^{2N}$.*

Proof. Let λ denote an eigenvalue of $\mathbf{\Gamma}$ and consider the characteristic equation $\det(\mathbf{\Gamma}-\lambda\mathbf{I}_{2N}) = 0$, which can be written in block-wise fashion as:

$$\det \begin{bmatrix} (1+\beta-\lambda)\mathbf{I}_N & -\beta\mathbf{I}_N \\ \mathbf{I}_N & -\lambda\mathbf{I}_N \end{bmatrix} = 0. \tag{10}$$

Now, we claim that $\lambda \neq 1+\beta$ and thus the block $(1+\beta-\lambda)\mathbf{I}_N$ in (10) is invertible. To see this, suppose on the contrary that $\lambda = 1+\beta$ and (10) holds. In this case, we have:

$$\det \begin{bmatrix} \mathbf{0}_N & -\beta\mathbf{I}_N \\ \mathbf{I}_N & -(1+\beta)\mathbf{I}_N \end{bmatrix} = \det(\beta\mathbf{I}_N) = -\beta^N \neq 0,$$

contradicting to the assumption that (10) holds. Now, since the block $(1+\beta-\lambda)\mathbf{I}_N$ is invertible, it follows from the Schur complements determinantal formulae [29] that

$$\begin{aligned} (10) =& \det\left[(1+\beta-\lambda)\mathbf{I}_N\right] \det\left[(-\lambda\mathbf{I}_N) - \mathbf{I}_N((1+\beta-\lambda)\mathbf{I}_N)^{-1}(-\beta\mathbf{I}_N)\right] \\ =& (1+\beta-\lambda)^N \det\left[\Big(-\lambda + \frac{\beta}{1+\beta-\lambda}\Big)\mathbf{I}_N\right] \\ =& [\lambda^2 - (1+\beta)\lambda + \beta]^N = (\lambda-1)^N(\lambda-\beta)^N = 0. \end{aligned} \tag{11}$$

Hence, the result stated in the lemma follows and this completes the proof. □

It is worth pointing out that the use of the matrix $\mathbf{\Gamma}$ in subsequent analysis and its eigen-spectrum property in Lemma 1 are new and has not appeared in the heavy-ball literature.

Our next key step toward proving Theorem 1 is to establish the following mean weight deviation bound:

Theorem 5 (Mean weight deviation bound)**.** *Under Algorithm 1 and a given K, there exists a constant C that depends on L, $s^{\max}$, and $a^{\max}$, such that $\mathbb{E}\{\|\mathbf{w}_{(K)}^{\infty} - \mathbf{w}_{(K)}^{*}\|\} \leq C\sqrt{K}$, where $\mathbf{w}_{(K)}^{\infty}$ denotes the weights $\mathbf{w}_K[t]$ under parameter K in steady-state.*

Proof. We start by rewriting the heavy-ball weight update equation in (7) in the following equivalent vector form:

$$\mathbf{w}_{(K)}[t+1] = \mathbf{w}_{(K)}[t] + (\mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}^{(1)}[t]) + \beta(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]) + \mathbf{u}^{(2)}[t], \tag{12}$$

where $\mathbf{u}^{(1)}[t]$ and $\mathbf{u}^{(2)}[t]$ are both projection terms that are defined as follows:

$$\mathbf{u}^{(1)}[t] \triangleq \big(\mathbf{s}[t] - \mathbf{a}[t] - \mathbf{q}[t]\big)^+,$$
$$\mathbf{u}^{(2)}[t] \triangleq \big(\mathbf{s}[t] - \mathbf{a}[t] - \mathbf{w}[t] - \beta(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]) - \mathbf{u}^{(1)}[t]\big)^+.$$

Noting that $\mathbf{u}^{(1)}[t], \mathbf{u}^{(1)}[t] \geq 0$, we can define $\mathbf{u}[t] \triangleq \mathbf{u}^{(1)}[t] + \mathbf{u}^{(2)}[t] \geq 0$ and further rewrite (12) as:

$$\mathbf{w}_{(K)}[t+1] - \mathbf{w}^*_{(K)} = \mathbf{w}_{(K)}[t] - \mathbf{w}^*_{(K)} + \big(\mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}[t]\big) + \beta\big(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]\big). \tag{13}$$

Note that since the momentum term in (13) depends on two consecutive time-slots of memory $\mathbf{w}_{(K)}[t]$ and $\mathbf{w}_{(K)}[t-1]$, traditional techniques used in establishing similar mean distance bounds (see, e.g., [13, 16]) cannot be directly applied. To overcome this challenge, we define a $2N$-dimensional vector $\mathbf{z}[t]$ as follows:

$$\mathbf{z}[t] \triangleq \begin{bmatrix} \mathbf{w}_{(K)}[t] - \mathbf{w}^*_{(K)} \\ \mathbf{w}_{(K)}[t-1] - \mathbf{w}^*_{(K)} \end{bmatrix}. \tag{14}$$

Then, it can be readily verified that (13) can be rewritten in terms of $\mathbf{z}[t]$ as follows:

$$\mathbf{z}[t+1] = \mathbf{\Gamma}\mathbf{z}[t] + \begin{bmatrix} \mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}[t]] \\ \mathbf{0}_N \end{bmatrix}. \tag{15}$$

Consider the following quadratic Lyapunov function: $V(\mathbf{z}[t]) \triangleq \frac{1}{2}\|\mathbf{z}[t]\|^2$, which can be interpreted as a measure of the combine distance of $\mathbf{w}[t]$ and $\mathbf{w}[t-1]$ deviating from $\mathbf{w}^*_{(K)}$. Then, the conditional expectation of the one-slot Lyapunov drift of $V(\mathbf{z}[t])$ can be written as:

$$\mathbb{E}\big\{\Delta V(\mathbf{z}[t])\big|\mathbf{z}[t]\big\} \triangleq \frac{1}{2}\mathbb{E}\big\{\|\mathbf{z}[t+1]\|^2 - \|\mathbf{z}[t]\|^2\big|\mathbf{z}[t]\big\}. \tag{16}$$

Let $\mathbb{1}_{\mathcal{A}}(\mathbf{x})$ be the indicator function that takes value 1 if $\mathbf{x} \in \mathcal{A}$ and 0 otherwise. After some algebraic derivations and upper-bounding (see [30, Appendix A] for proof details), we arrive at the following result:

Proposition 1. *Let* $\mathbf{w}$ *be the first* N *entries in* $\mathbf{z}[t] = \mathbf{z}$*. Let* $B \triangleq \frac{N}{2}[A + (s^{\max})^2]$*. There exist constants* $\delta, \eta > 0$ *such that*

$$\mathbb{E}\{\Delta V(\mathbf{z}[t])|\mathbf{z}[t] = \mathbf{z}\} \leq -\frac{\delta}{\sqrt{K}}\big\|\mathbf{w} - \mathbf{w}^*_{(K)}\big\|\mathbb{1}_{\mathcal{B}^c}(\mathbf{w}) + \eta\mathbb{1}_{\mathcal{B}}(\mathbf{w}),$$

where $\mathcal{B} \triangleq \{\mathbf{w} : \big\|\mathbf{w} - \mathbf{w}^*_{(K)}\big\| < \sqrt{B\Phi K}\}$*, and* $\mathcal{B}^c$ *denotes the complement of* $\mathcal{B}$*.*

Now, we consider the T-step conditional mean Lyapunov drift. For notational simplicity, we define a set $\Omega \triangleq \{\mathbf{z} \in \mathbb{R}^{2N} : \|(\mathbf{z})_{1:N}\| \in \mathcal{B}\}$, where $(\mathbf{z})_{1:N}$ denotes the first N entries in $\mathbf{z}$ and $\mathcal{B}$ is as defined in Proposition 1. By telescoping (16) from $t = 0$ to T, we have that

$$\begin{aligned}
&\mathbb{E}\{V(\mathbf{z}[T])|\mathbf{z}[0]\} - V(\mathbf{z}[0]) = \sum_{t=0}^{T-1} \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[0]\} \\
&\overset{(a)}{=} \sum_{t=0}^{T-1} \int_{\mathbb{R}^{2N}} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[t] = \mathbf{z}\} d\mathbf{z} \\
&= \sum_{t=0}^{T-1} \int_{\Omega} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[t] = \mathbf{z}\} d\mathbf{z} + \\
&\quad \sum_{t=0}^{T-1} \int_{\Omega^c} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[t] = \mathbf{z}\} d\mathbf{z}, \qquad (17)
\end{aligned}$$

where (a) follows from the fact that $\mathbf{z}[t]$ is a continuous state Markov chain in $\mathbb{R}^{2N}$. It then follows from Proposition 1 that (17) can be upper-bounded as:

$$\mathbb{E}\{V(\mathbf{z}[T])|\mathbf{z}[0]\} - V(\mathbf{z}[0]) \leq -\frac{\delta}{\sqrt{K}} \int_{\Omega^c} \left(\|\mathbf{w}_{(K)}[t] - \mathbf{w}^*_{(K)}\| \sum_{t=0}^{T-1} p_{\mathbf{z}[t]|\mathbf{z}[0]} \right) d\mathbf{z} + \eta \int_{\Omega} \left(\sum_{t=0}^{T-1} p_{\mathbf{z}[t]|\mathbf{z}[0]} \right) d\mathbf{z}. \qquad (18)$$

Note that for any $\mathbf{z} \in \mathbb{R}^{2N}$, $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} p_{\mathbf{z}[t]|\mathbf{z}[0]} = p^{\infty}_{\mathbf{z}}$ for all $\mathbf{z}[0]$, where $p^{\infty}_{\mathbf{z}}$ denotes the stationary distribution of the continuous state Markov chain $\mathbf{z}[t]$. Moving $V(\mathbf{z}[0])$ to the right hand side (RHS) of (18), dividing both sides of (18) by T, and letting $T \to \infty$ yields:

$$0 \leq -\frac{\delta}{\sqrt{K}} \int_{\Omega^c} p^{\infty}_{\mathbf{z}} \|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\| d\mathbf{z} + \eta \int_{\Omega} p^{\infty}_{\mathbf{z}} d\mathbf{z}. \qquad (19)$$

Rearranging terms and adding $\frac{\delta}{\sqrt{K}} \int_{\Omega} p^{\infty}_{\mathbf{z}} \|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\|$ yields:

$$\begin{aligned}
&\frac{\delta}{\sqrt{K}} \int_{\mathbb{R}^{2N}} p^{\infty}_{\mathbf{z}} \|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\| d\mathbf{z} \\
&\leq \int_{\Omega} \left(\eta + \frac{\delta}{\sqrt{K}} \|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\| \right) p^{\infty}_{\mathbf{z}} d\mathbf{z} \\
&\overset{(a)}{\leq} (\eta + \delta\sqrt{B\Phi}) \int_{\Omega} p^{\infty}_{\mathbf{z}} d\mathbf{z} \\
&\leq \eta + \delta\sqrt{B\Phi}, \qquad (20)
\end{aligned}$$

where (a) follows from the definitions of Ω and $\mathcal{B}$. Note that the left-hand-side (LHS) of (20) is exactly $\frac{\delta}{\sqrt{K}} \mathbb{E}\{\|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\|\}$. Finally, multiplying both sides of (20) by $\frac{\sqrt{K}}{\delta}$ yields:

$$\mathbb{E}\{\|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\|\} \leq \left(\frac{\eta}{\delta} + \sqrt{B\Phi} \right) \sqrt{K} = O(\sqrt{K}), \qquad (21)$$

i.e., the result stated in Theorem 5. This completes the proof. □

Proof of Theorem 1. Consider the heavy-ball update, which can be written in the following form:

$$\mathbf{w}_{(K)}[t+1] = \mathbf{w}_{(K)}[t] + \Delta\mathbf{q}[t] + \beta(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]) + \mathbf{u}^{(2)}[t].$$

Rearranging terms and noting that $\mathbf{u}^{(2)}[t] \geq \mathbf{0}$, we have

$$\Delta\mathbf{q}[t] \leq \big(\mathbf{w}_{(K)}[t+1] - \mathbf{w}_{(K)}[t]\big) - \beta\big(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]\big). \tag{22}$$

Telescoping the inequality in (22) from $t = 0$ to $T-1$ yields:

$$\begin{aligned}\sum_{t=0}^{T-1} \Delta\mathbf{q}[t] \leq& \big(\mathbf{w}_{(K)}[T] - \mathbf{w}_{(K)}[0]\big) - \beta\big(\mathbf{w}_{(K)}[T-1] - \mathbf{w}_{(K)}[-1]\big) \\ =& \mathbf{w}_{(K)}[T] - \beta\mathbf{w}_{(K)}[T-1],\end{aligned} \tag{23}$$

where the last equality holds because, by assumption, $\mathbf{w}_{(K)}[0] = \mathbf{w}_{(K)}[-1] = \mathbf{0}$. Also, since $\mathbf{q}[0] = \mathbf{0}$, we have

$$\|\mathbf{q}[T]\|_1 = \|\mathbf{q}[0] + \sum_{t=0}^{T-1} \Delta\mathbf{q}[t]\|_1 \leq \|\mathbf{w}_{(K)}[T] - \beta\mathbf{w}_{(K)}[T-1]\|_1.$$

Taking expectation on both sides, letting $T \to \infty$, and taking limits yields:

$$\begin{aligned}&\limsup_{T\to\infty} \mathbb{E}\big\{\|\mathbf{q}[T]\|_1\big\} \leq \mathbb{E}\big\{\mathbf{w}^{\infty}_{(K)} - \beta\mathbf{w}^{\infty}_{(K)}\big\} \\ \overset{(a)}{\leq}& \mathbf{w}^{*}_{(K)} + O(\sqrt{K}) - \beta\big(\mathbf{w}^{*}_{(K)} - O(\sqrt{K})\big) \\ =& (1-\beta)\mathbf{w}^{*}_{(K)} + (1+\beta)O(\sqrt{K}),\end{aligned} \tag{24}$$

where (a) follows from Theorem 5 and $\|\cdot\|_1 \leq \sqrt{N}\|\cdot\|$. Moreover, in the asymptotic regime where β approaches 1 at a rate equal or faster than $1-\beta = O(\frac{1}{\sqrt{K}})$, it follows from (24) that

$$\limsup_{T\to\infty} \mathbb{E}\big\{\|\mathbf{q}[t]\|_1\big\} \approx O(\sqrt{K}). \tag{25}$$

That is, the delay growth at $O(\sqrt{K})$. This completes the proof. □

Proof of Theorem 2. We first prove the optimality gap result for the mean of the stationary rates $a^{\infty}_{(K),n} \triangleq \mathbb{E}\{\min\{U_n'^{-1}(\frac{w^{\infty}_{(K),n}}{K}), a^{\max}\}\}$. Note that $\mathbb{E}\{a_n[t]|w_{(K),n}[t]\} = \min\{U_n'^{-1}(\frac{w_{(K),n}[t]}{K}), a^{\max}\}$

and $a_n^* = U_n'^{-1}(\frac{w^*_{(K),n}}{K})$, $\forall n$. Thus, we have

$$\begin{aligned}
\left\|\mathbf{a}^{\infty}_{(K)} - \mathbf{a}^*\right\|^2 &= \sum_{n=1}^{N}\left[\mathbb{E}\left\{\min\left\{U_n'^{-1}\left(\frac{w^{\infty}_{(K),n}}{K}\right), M\right\}\right\} - U_n'^{-1}\left(\frac{w^*_{(K),n}}{K}\right)\right]^2 \\
&\overset{(a)}{\leq} \sum_{n=1}^{N}\mathbb{E}\left\{\left[\min\left\{U_n'^{-1}\left(\frac{w^{\infty}_{(K),n}}{K}\right), M\right\} - U_n'^{-1}\left(\frac{w^*_{(K),n}}{K}\right)\right]^2\right\} \\
&\overset{(b)}{\leq} \sum_{n=1}^{N}\mathbb{E}\left\{\left[U_n'^{-1}\left(\frac{w^{\infty}_{(K),n}}{K}\right) - U_n'^{-1}\left(\frac{w^*_{(K),n}}{K}\right)\right]^2\right\} \\
&\overset{(c)}{=} \sum_{n=1}^{N}\mathbb{E}\left\{\left[\left[U_n'^{-1}\left(\frac{\tilde{w}_{(K),n}}{K}\right)\right]'\left(\frac{w^{\infty}_{(K),n}}{K} - \frac{w^*_{(K),n}}{K}\right)\right]^2\right\} \\
&\overset{(d)}{=} \sum_{n=1}^{N}\mathbb{E}\left\{\left[\left[\frac{1}{U_n''\left(\frac{\tilde{w}_{(K),n}}{K}\right)}\right]^2\left(\frac{w^{\infty}_{(K),n}}{K} - \frac{w^*_{(K),n}}{K}\right)\right]^2\right\} \\
&\overset{(e)}{\leq} \sum_{n=1}^{N}\mathbb{E}\left\{\frac{1}{\phi^2}\left(w^{\infty}_{(K),n} - w^*_{(K),n}\right)^2\frac{1}{K^2}\right\} \\
&\overset{(f)}{=} \frac{1}{\phi^2 K^2}\mathbb{E}\left\{\sum_{n=1}^{N}\left(w^{\infty}_{(K),n} - w^*_{(K),n}\right)^2\right\} \\
&= \frac{1}{\phi^2 K^2}\mathbb{E}\left\{\|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\|^2\right\}, \qquad (26)
\end{aligned}$$

where (a) follows from the convexity of quadratic function and Jensen's inequality; (b) follows from the non-expansion property of the $\min\{\cdot\}$ function; (c) follows by using mean value theorem for some $\tilde{w}_{(K),n} \in \left[\min\{w^{\infty}_{(K),n}, w^*_{(K),n}\}, \max\{w^{\infty}_{(K),n}, w^*_{(K),n}\}\right]$; (d) follows from inverse function lemma; (e) follows from the strong convexity assumption in (3); and (f) follows from exchanging the order or summation and expectation.

Now, consider the term $\mathbb{E}\left\{\|\mathbf{w}^{\infty}_{(K)} - \mathbf{w}^*_{(K)}\|^2\right\}$ in (26). From the proof of Proposition 1, we have (cf. [30, Appendix A, Eq. (51)]), we have the following one-slot mean Lyapunov drift bound:

$$\mathbb{E}\{\Delta V(\mathbf{z}[t])|\mathbf{z}[t]\} \leq -\frac{1}{\Phi K}\left\|\mathbf{w}_{(K)}[t] - \mathbf{w}^*_{(K)}\right\|^2 + B. \qquad (27)$$

Following the same argument in the proof of Theorem 5, we telescope the inequality in (27) from

$t = 0$ to $T-1$ to obtain:

$$\mathbb{E}\{V(\mathbf{z}[T])|\mathbf{z}[0]\} - V(\mathbf{z}[0]) = \sum_{t=0}^{T-1} \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[0]\}$$
$$= \sum_{t=0}^{T-1} \int_{\mathbb{R}^{2N}} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \mathbb{E}\{V(\mathbf{z}[t+1]) - V(\mathbf{z}[t])|\mathbf{z}[t] = \mathbf{z}\} d\mathbf{z}$$
$$= \sum_{t=0}^{T-1} \int_{\mathbb{R}^{2N}} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \mathbb{E}\{\Delta V(\mathbf{z}[t])|\mathbf{z}[t]\} d\mathbf{z}$$
$$\leq -\frac{1}{\Phi K} \sum_{t=0}^{T-1} \int_{\mathbb{R}^{2N}} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \left\|\mathbf{w} - \mathbf{w}^*_{(K)}\right\|^2 d\mathbf{z} + TB. \tag{28}$$

Dividing both sides of (28) by $\frac{T}{LK}$, rearranging terms, and letting $T \to \infty$, we have

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \int_{\mathbb{R}^{2N}} p_{\mathbf{z}[t]|\mathbf{z}[0]}(\mathbf{z}) \left\|\mathbf{w} - \mathbf{w}^*_{(K)}\right\|^2 d\mathbf{z} \leq B\Phi K. \tag{29}$$

Note that the LHS of (29) is precisely $\mathbb{E}\left\{\|\mathbf{w}^\infty_{(K)} - \mathbf{w}^*_{(K)}\|^2\right\}$. Hence, it follows that

$$\|\mathbf{a}^\infty_{(K)} - \mathbf{a}^*\|^2 \leq \frac{1}{\phi^2 K^2} \mathbb{E}\left\{\|\mathbf{w}^\infty_{(K)} - \mathbf{w}^*_{(K)}\|^2\right\} \leq \frac{B\Phi}{\phi^2} \frac{1}{K}. \tag{30}$$

Taking square root on both sides of (30) yields:

$$\|\mathbf{a}^\infty_{(K)} - \mathbf{a}^*\| \leq \frac{\sqrt{B\Phi}}{\phi} \frac{1}{\sqrt{K}} = O(\frac{1}{\sqrt{K}}),$$

i.e., the proof of the first half of Theorem 2 is complete.

Next, we proof the optimality gap result for the objective value, i.e., $U\left(\mathbf{a}^\infty_{(K)}\right) \geq U(\mathbf{a}^*) - O(1/K)$. To this end, similar to the proof of Theorem 5, we define an augmented vector and its quadratic Lyapunov function as follows:

$$\mathbf{y}[t] \triangleq \begin{bmatrix} \mathbf{w}_{(K)}[t] \\ \mathbf{w}_{(K)}[t-1] \end{bmatrix} \quad \text{and} \quad L(\mathbf{y}[t]) = \frac{1}{2}\|\mathbf{y}[t]\|^2.$$

Following the same steps in the proof of Theorem 5, one can verify that

$$\mathbf{y}[t+1] = \mathbf{\Gamma}\mathbf{y}[t] + \begin{bmatrix} \mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}[t] \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{\Gamma}$ is the same as defined in (9). Then, following the same steps as in the proof of Proposition 1, we can show that the conditional expectation of the one-slot Lyapunov drift can be bounded as follows:

$$\mathbb{E}\left\{\Delta L(\mathbf{y}[t])\middle|\mathbf{y}[t]\right\} \leq \mathbb{E}\left\{\frac{1}{2}\|\mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}[t]\|^2 + \langle \mathbf{a}[t] - \mathbf{s}[t] + \mathbf{u}[t], \mathbf{y}[t]\rangle \middle|\mathbf{y}[t]\right\}$$
$$\leq -\left(\mathbf{w}_{(K)}[t]\right)^\top \mathbb{E}\{\mathbf{a}[t] - \mathbf{s}[t]\big|\mathbf{y}[t]\} + B. \tag{31}$$

Note that (31) is in the same form as in [12, Eq. (24)]. Then, following the same arguments in [12], we have that $U\left(\mathbf{a}^\infty_{(K)}\right) \geq U(\mathbf{a}^*) - O(1/K)$. This completes the proof. □

Proof of Theorem 3. We first show the ranges of K and β that suffice for convergence. Due to the one-to-one mapping between $\mathbb{E}\{\mathbf{a}[t]|\mathbf{w}_{(K)}[t]\}$ and $\mathbf{w}_{(K)}[t]$, the convergence of $\mathbb{E}\{\mathbf{a}[t]|\mathbf{w}_{(K)}[t]\}$ can be equivalently done by examining the convergence fo $\mathbf{w}_{(K)}[t]$. Note that (7) can be written as:

$$\mathbf{w}_{(K)}[t+1] \leq \mathbf{w}_{(K)}[t] + (\mathbf{a}[t] - \mathbf{s}[t]) + \beta(\mathbf{w}_{(K)}[t] - \mathbf{w}_{(K)}[t-1]).$$

Dividing both sides by K (scaling does not affect the convergence), we have:

$$\mathbf{w}_{(1)}[t+1] \leq \mathbf{w}_{(1)}[t] + \frac{1}{K}(\mathbf{a}[t] - \mathbf{s}[t]) + \beta(\mathbf{w}_{(1)}[t] - \mathbf{w}_{(1)}[t-1]).$$

Note that the right-hand-side is in the same form as in the classical unconstrained heavy-ball method (cf. [21, 26]) with step-size being $\frac{1}{K}$. Hence, following [21, Chap. 3.2, Theorem 1], we have the sufficient condition for convergence as follows: $\frac{1}{K} \in (0, \frac{(1+\beta)}{\Phi}]$, and $\beta \in [0, 1)$. After some manipulations and noting that $\beta > 0$, we arrive at $K \in (\frac{\Phi}{4}, \infty]$ and $\beta \in [\max\{0, \frac{\Phi}{2K} - 1\}, 1)$, i.e., the result stated in Theorem 3. Also, the convergence factor upper-bound in Theorem 3 follows directly from [26, Theorem 1]. The minimum upper-bound R^* and the minimizers K^* and β^* follow from [21, Chap. 3.2, Theorem 1]. This completes the proof. □

Proof of Theorem 4. The results of utility and delay scalings simply follow from Theorems 2 and 1, respectively. The convergence time scaling result follows from plugging in the convergence factor upper-bound from Theorem 3 into the convergence time definition in [30, Eq. (13)], inverting the convergence rate expression, and noting from $\|\mathbf{w}_{(K)}[t] - \mathbf{w}^*_{(K)}\| = O(1/\sqrt{K})$. This completes the proof. □

List of Symbols, Abbreviations, and Acronyms

AFRL	Air Force Research Laboratory
API	Application Programming Interface
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CORE	Common Open Research Emulator
CSI	Channel State Information
CSMA	Carrier-Sense Multiple Access
EMANE	Extendable Mobile Ad-hoc Network Emulator
GIG	Global Information Grid
GUI	Graphical User Interface
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
MGEN	Multi-Generator
OS	Operating System
PHY	Physical
PI	Principal Investigator
SNR	Signal to Noise Ratio
TDMA	Time Division Multiple Access
UAS	Unmanned Aerial System