Optimal Control for Generalized Network-Flow Problems

Abhishek Sinha, Eytan Modiano
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139
Email: sinhaa@mit.edu, modiano@mit.edu

***Abstract*—We consider the problem of throughput-optimal packet dissemination, in the presence of an arbitrary mix of unicast, broadcast, multicast and anycast traffic, in a general wireless network. We propose an online dynamic policy, called Universal Max-Weight (UMW), which solves the above problem efficiently. To the best of our knowledge, UMW is the first throughput-optimal algorithm of such versatility in the context of generalized network flow problems. Conceptually, the UMW policy is derived by relaxing the precedence constraints associated with multi-hop routing, and then solving a min-cost routing and max-weight scheduling problem on a *virtual network of queues*. When specialized to the unicast setting, the UMW policy yields a throughput-optimal *cycle-free routing* and link scheduling policy. This is in contrast to the well-known throughput-optimal *Back-Pressure* (BP) policy which allows for packet cycling, resulting in excessive delay. Extensive simulation results show that the proposed policy incurs a substantially lower delay as compared to the BP policy. The proof of throughput-optimality of the UMW policy combines techniques from stochastic Lyapunov theory with a sample path argument from *adversarial queueing theory* and may be of independent theoretical interest.**

I. Introduction

The *Generalized Network Flow* problem refers to efficient transportation of messages, generated at source node(s), to a set of designated destination node(s) over a multi-hop network. Depending on the number of destination nodes associated with each source node, the problem is known either as *unicast* (single destination node), *broadcast* (all node are destination nodes), *multicast* (some nodes are destination nodes) or *anycast* (several choices for a single destination node). Over the last few decades, a tremendous amount of research effort has been directed to address each of the above problems in different networking contexts. However, despite the increasingly diverse mix of internet traffic, to the best of our knowledge, there exists no universal solution to the general problem, only isolated solutions that do not interoperate and are often suboptimal. In this paper, we provide the first such universal solution: A throughput optimal dynamic control policy for the generalized network flow problem.

We start with a brief discussion of the above networking problems and then survey the relevant literature.

In the **Broadcast** problem, packets generated at a source need to be distributed among all nodes in the network. In the classic paper of Edmonds [1], the broadcast capacity of a wired network is derived and an algorithm is proposed to compute the set of maximum number of edge-disjoint spanning trees, which together achieve the maximum broadcast throughput. The resulting algorithm is combinatorial in nature and does not have a wireless counterpart, with associated interference-free edge activations. Following Edmonds' work, a variety of different broadcast algorithms have been proposed in the literature, each one targeted to optimize different metrics, such as, delay [2], energy consumption [3] and fault-tolerance [4]. In the context of optimizing throughput, the paper [5] proposes a randomized broadcast policy, which is optimal for wired networks. However, extending this algorithm to the wireless setting proves to be difficult [6]. In the paper [7], we propose an optimal broadcast algorithm for a general wireless network, albeit with exponential complexity. In addition to this, in a recent series of papers [8] [9], we propose a simple throughput-optimal broadcast algorithm for wireless networks with an underlying DAG topology. However, this algorithm does not extend to non-DAG networks.

The **Multicast** problem is a generalization of the broadcast problem, in which the packets generated at a source node needs to be efficiently distributed to a subset of nodes in the network. In its combinatorial version, the multicast problem reduces to finding the maximum number of edge-disjoint trees, spanning the source node and destination nodes. This problem is known as the *Steiner Tree Packing* problem, which is NP-hard [10]. Numerous algorithms have been proposed in the literature for solving the multicast problem. In [11] [12], back-pressure type algorithms are proposed for multicasting over wired and wireless networks respectively. These algorithms forward packet over a set of pre-computed distribution trees and are limited to the throughput obtainable by those trees. Moreover, computing and maintaining these trees is impractical in large and time-varying networks. Note that, because of packet duplications, the Multicast and Broadcast problems do not satisfy standard flow conservation constraints, and thus the design of throughput-optimal algorithms is non-trivial.

The **Unicast** problem involves a single source and a single destination. The celebrated Back-Pressure (BP) algorithm [13] was proposed for the unicast problem. In this algorithm, the routing and scheduling decisions are taken based on local queue length differences. As a result, BP explores all possible paths for routing and usually takes a long time for convergence, resulting in considerable queueing delay, especially in lightly loaded networks. Subsequently, a number of refinements have been proposed to improve the delay

This work was supported by the NSF Grants CNS-1217048 and CNS-1524317.

characteristics of the BP algorithm. In the paper [14], the BP algorithm is combined with hop length based shortest path routing for faster route discovery, and the paper [15] proposes a second order algorithm to improve delay.

The **Anycast** problem involves routing from a single source to *any one* of several feasible destinations. Anycast is increasingly used in Content-Distribution Networks (CDNs) for optimally distributing geo-replicated contents [16].

Our proposed solution uses a *virtual network of queues* - one virtual queue per link in the network. We solve the routing problem dynamically by using a "weighted-shortest-route" computation on the virtual network and then using the corresponding route on the physical network. Optimal link scheduling is performed by a max-weight activation, also in the virtual network, and then using the resulting activation in the physical network. The overall algorithm is dynamic, cycle-free, and solves the generalized routing and scheduling problem optimally (i.e., maximally stable or throughput-optimal). In addition to this, the proposed **UMW** policy offers the following advantages:

1) **Generalized Solution:** Unlike the BP policy, which solves only the unicast problem, the proposed **UMW** policy efficiently addresses all of the aforementioned network flow problems in both wired and wireless networks in a very general setting.
2) **Delay Reduction:** Although the celebrated BP policy is throughput-optimal, its average delay performance is known to be poor due to the occurrence of packet-cycling in the network [14] [17]. In our proposed **UMW** policy, each packet traverses a dynamically selected *acyclic* route, which drastically reduces the average latency.
3) **State-Complexity Reduction:** Unlike the BP policy, which maintains *per-flow* queues at each node, the proposed **UMW** policy maintains only a virtual queue counter and a priority-queue per link, irrespective of the number and type of flows in the network. This reduces the amount of overhead that needs to be maintained for efficient operation.
4) **Efficient Implementation:** In the BP policy, routing decisions are made hop-by-hop by the intermediate nodes. This puts a considerable amount of computational overhead on the individual nodes. In contrast, in the proposed **UMW** policy, the entire route of a packet is determined at its source (similar to *dynamic source-routing* [18]). Hence, the entire computational requirement is transferred to the source, which often has higher computational/energy resources than the nodes in the rest of the network (*e.g.*, wireless sensor networks).

The rest of the paper is organized as follows: In section II we discuss the basic system model and formulate the problem. In section III we give a brief overview of the proposed **UMW** policy. In section IV we discuss the structure and dynamics of the virtual queues, on which **UMW** is based. In section V we prove its stability property in the multi-hop physical network. In section VI we discuss implementation details. In section VII we provide extensive simulation results, comparing **UMW** with other competing algorithms. In section VIII we conclude the paper with directions for further research. Due to space limitations, the details of technical proofs of theorems have been provided in the technical report [19].

II. System Model and Problem Formulation

A. Network Model

We consider a wireless network with arbitrary topology, represented by the graph $\mathcal{G}(V,E)$. The network consists of $|V| = n$ nodes and $|E| = m$ links. Time is slotted. A link, if activated, can transmit one packet per slot. Due to wireless interference constraints, only certain subsets of links may be activated together at any slot. The set of all admissible link-activations is known as the *activation set*, and is denoted by $\mathcal{M} \subseteq 2^E$. We do not impose any restriction on the structure of the activation set $\mathcal{M}$. As an example, in the case of *node-exclusive* or primary interference constraints [20], the activation set $\mathcal{M}_{\text{primary}}$ consists of the set of all *matchings* [21] in the graph $\mathcal{G}(V,E)$. Wired networks are a special case of the above model, where the activation set $\mathcal{M}_{\text{wired}} = 2^E$. In other words, in wired networks, packets can be transmitted over all links simultaneously.

B. Traffic Model

In this paper, we consider the *Generalized Network Flow problem*, where incoming packets at a source node are to be distributed among an arbitrary set of destination nodes in a multi-hop fashion. Formally, the set of all distinct classes of incoming traffic is denoted by $\mathcal{C}$. A class c traffic is identified by its source node $s^{(c)} \in V$ and the set of its required destination nodes $\mathcal{D}^{(c)} \subseteq V$. As explained below, by varying the structure of the destination set $\mathcal{D}^{(c)}$ of a class c, this general framework yields the following four fundamental flow problems as special cases:

- **Unicast**: All class c packets, arriving at a source node $s^{(c)}$, are required to be delivered to a single destination node $\mathcal{D}^{(c)} = \{t^{(c)}\}$.
- **Broadcast**: All class c packets, arriving at a source node $s^{(c)}$, are required to be delivered to all nodes in the network, i.e., $\mathcal{D}^{(c)} = V$.
- **Multicast**: All class c packets, arriving at a source node $s^{(c)}$, are required to be delivered to a proper subset of nodes $\mathcal{D}^{(c)} = \{t_1^{(c)}, t_2^{(c)}, \ldots, t_k^{(c)}\} \subsetneq V$.
- **Anycast**: A Packet of class c, arriving at a source node $s^{(c)}$, is required to be delivered to *any one* of a given set of k nodes $\mathcal{D}^{(c)} = t_1^{(c)} \oplus t_2^{(c)} \oplus \ldots \oplus t_k^{(c)}$. Thus the anycast problem is similar to the unicast problem, with all destinations forming a single *super* destination-node.

Arrivals are assumed to be i.i.d. at every slot, with $A^{(c)}(t)$ packets from class c arriving at the source node $s^{(c)}$ at slot t. The mean rate of arrival for class c is $\mathbb{E}A^{(c)}(t) = \lambda^{(c)}$. The arrival rate vector is given by $\boldsymbol{\lambda} = \{\lambda^{(c)}, c \in \mathcal{C}\}$. Total number of external packet arrivals to the entire network at any slot t is assumed to be bounded by a finite number $A_{\max}$.

C. Policy-Space

An admissible policy π for the generalized network flow problem executes the following two actions at every slot t:

- Link Activations: Activating a subset of interference-free links $\boldsymbol{s}(t)$ from the activation set $\mathcal{M}$.
- Packet Duplications and Forwarding: Possibly duplicating [1] and forwarding packets over the activated links. Due to the link capacity constraint, at most one packet may be transmitted over an active link per slot.

The set of all admissible policies is denoted by Π. The set Π is unconstrained otherwise and includes policies which may use all past and future packet arrival information.
A policy $\pi \in \Pi$ is said to *support an arrival rate-vector* $\boldsymbol{\lambda}$ if, under the action of the policy π, the destination nodes of any class c receive distinct class c packets at the rate $\lambda^{(c)}, c \in \mathcal{C}$. Formally, let $R^{(c)}(t)$ denote the number of distinct class-c packets, received in common by all destination nodes $i \in \mathcal{D}^{(c)}$, under the action of the policy π, up to time t[2].

Definition 1. [Policy Supporting Rate-Vector $\boldsymbol{\lambda}$]*: A policy $\pi \in \Pi$ is said to support an arrival rate vector $\boldsymbol{\lambda}$ if*

$$\liminf_{t\to\infty} \frac{R^{(c)}(t)}{t} = \lambda^{(c)}, \quad \forall c \in \mathcal{C}, \text{ w.p.1} \tag{1}$$

The network-layer capacity region $\boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$ [3] is defined to be the set of all supportable rates, i.e.,

$$\boldsymbol{\Lambda}(\mathcal{G},\mathcal{C}) \stackrel{\text{def}}{=} \{\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|} : \exists \pi \in \Pi \text{ supporting } \boldsymbol{\lambda}\} \tag{2}$$

Clearly, the set $\boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$ is *convex* (using the usual *time-sharing* argument). A policy $\pi^* \in \Pi$, which supports any arrival rate $\boldsymbol{\lambda}$ in the interior of the capacity-region $\boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$, is called a *throughput-optimal* policy.

D. Admissible Routes of Packets

We will design a throughput-optimal policy, which delivers a packet p to any node in the network *at most* once.[4] This immediately implies that the set of all admissible routes $\mathcal{T}^{(c)}$ for packets of any class c, in general, comprises of trees rooted at the corresponding source node $s^{(c)}$. In particular, depending on the type of class c traffic, the topology of the admissible routes $\mathcal{T}^{(c)}$ takes the following special forms:

- **Unicast Traffic**: $\mathcal{T}^{(c)}$ = set of all $s^{(c)} - t^{(c)}$ paths in the graph $\mathcal{G}$.
- **Broadcast Traffic**: $\mathcal{T}^{(c)}$ = set of all spanning trees in the graph $\mathcal{G}$, rooted at $s^{(c)}$.
- **Multicast Traffic**: $\mathcal{T}^{(c)}$ = set of all Steiner trees [10] in $\mathcal{G}$, rooted at $s^{(c)}$ and spanning the vertices $\mathcal{D}^{(c)} = \{t_1^{(c)}, t_2^{(c)}, \ldots, t_k^{(c)}\}$.
- **Anycast Traffic**: $\mathcal{T}^{(c)}$ = union of all $s^{(c)} - t_i^{(c)}$ paths in the graph $\mathcal{G}$, $i = 1, 2, \ldots, k$.

E. Characterization of the Network-Layer Capacity Region

Consider any arrival vector $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$. By definition, there exists an admissible policy $\pi \in \Pi$, which supports the arrival rate $\boldsymbol{\lambda}$ by means of storing, duplicating and forwarding packets efficiently. Taking time-averages over the actions of the policy π, it is clear that there exists a *randomized* flow-decomposition and scheduling policy to route the packets such that none of the edges in the network is overloaded. Indeed, in the following theorem, we show that for every $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$, there exist non-negative scalars $\{\lambda_i^{(c)}\}$, indexed by the admissible routes $T_i^{(c)} \in \mathcal{T}^{(c)}$ and a convex combination of the link-activation vectors $\overline{\boldsymbol{\mu}} \in \text{conv}(\mathcal{M})$ such that,

$$\lambda^{(c)} = \sum_{T_i^{(c)} \in \mathcal{T}^{(c)}} \lambda_i^{(c)}, \quad \forall c \in \mathcal{C} \tag{3}$$

$$\lambda_e \stackrel{\text{(def.)}}{=} \sum_{(i,c):e\in T_i^{(c)}, T_i^{(c)}\in\mathcal{T}^{(c)}} \lambda_i^{(c)} \leq \overline{\mu}_e, \quad \forall e \in E. \tag{4}$$

Eqn. (3) denotes decomposition of the average incoming flows into different admissible routes and Eqn. (4) denotes the fact that none of the edges in the network is overloaded, i.e. arrival rate of packets to any edge e under the policy π is *at most* the rate allocated by the policy π to the edge e to serve packets. To state the result precisely, define the set $\overline{\boldsymbol{\Lambda}}$ to be the set of all arrival vectors $\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}|}$, for which there exists a randomized activation vector $\overline{\boldsymbol{\mu}} \in \text{conv}(\mathcal{M})$ and a non-negative flow decomposition $\{\lambda_i^{(c)}\}$, such that Eqns. (3) and (4) are satisfied. We have the following theorem:

Theorem 1. *The network-layer capacity region $\boldsymbol{\Lambda}(\mathcal{G},\mathcal{C})$ is characterized by the set $\overline{\boldsymbol{\Lambda}}$, up to its boundary.*

Proof of Theorem 1 consists of two parts: converse and achievability. Proof of the *converse* is given in Appendix A of [19], where we show that all supportable arrival rates must belong to the set $\overline{\boldsymbol{\Lambda}}$. The main result of this paper, developed in the subsequent sections, is the construction of an efficient admissible policy, called **U**niversal **M**ax-**W**eight (**UMW**), which *achieves* any arrival-rate in the interior of the set $\overline{\boldsymbol{\Lambda}}$.

[1]In order to transmit a packet over multiple downstream links (*e.g.* in Broadcast or Multicast), the sender must duplicate the packet and send the copies to the respective downstream link buffers.

[2]To be precise, any one of the destination nodes for Anycast.

[3]Note that, Network-layer capacity region is, in general (e.g. multicast), different from the Information-Theoretic capacity region [22].

[4]This should be contrasted with the popular throughput-optimal unicast policy *Back-Pressure* [13], which does not satisfy this constraint and may deliver the same packet to a node multiple times, thus potentially degrading its delay performance.

III. Overview of the **UMW** Policy

In this section, we present a brief overview of the operation of the **UMW** policy, designed and analyzed in the subsequent sections. Central to the **UMW** policy is a global state-vector, called virtual queues $\tilde{\boldsymbol{Q}}(t)$, used for packet routing and link activations. Each component of the virtual queues is updated at every slot according to a one-hop queueing (Lindley) recursion, corresponding to a *relaxed* network, described in detail in section IV. Unlike the well-known Back-Pressure algorithm for the unicast problem [13], in which packet routing decisions are made hop-by-hop using physical queue lengths $\boldsymbol{Q}(t)$, the **UMW** policy prescribes an admissible route to each incoming packet *immediately upon its arrival*. This route selection decision is dynamically made by solving a suitable min-cost routing problem (e.g., shortest path, MST etc.) at the source with edge costs given by the current virtual queue length vector $\tilde{\boldsymbol{Q}}(t)$. Link-activation decisions at each slot are made by a Max-Weight algorithm with link-weights set equal to $\tilde{\boldsymbol{Q}}(t)$. Having fixed the routing and activation policy as above, in section V we design a packet scheduling algorithm for the physical network, which efficiently resolves contention among multiple packets that wait to cross the same (active) edge at the same slot in the physical network. We show that the overall policy is throughput-optimal. One significantly new feature of our algorithm is that, unlike BP, this algorithm is entirely oblivious to the length of the physical queues of the network and utilizes the auxiliary virtual queue lengths for stabilizing the former.
Our proof of throughput-optimality of **UMW** leverages ideas from *deterministic* adversarial queuing theory and combines it effectively with the *stochastic* Lyapunov-drift based techniques and may be of independent theoretical interest.

IV. Global Virtual Queues: Structures, Algorithms, and Stability

In this section, we introduce the notion of *virtual queues*, which is obtained by *relaxing* the dynamics of the physical queues of the network in the following fashion [5].

A. Precedence Constraints

In a multi-hop network, if a packet p is being routed along the path $T = l_1 - l_2 - \ldots - l_k$, where $l_i \in E$ is the i^{th} link on its path, then by the principle of causality, the packet p cannot be physically transmitted over the j^{th} link l_j if it has not already been transmitted by the first $j-1$ links $l_1, l_2, \ldots, l_{j-1}$. This constraint is known as the *precedence constraint* in the network scheduling literature [24]. In the following, we make a radical departure by relaxing this constraint to obtain a simpler single-hop virtual system, which will play a key role in designing our policy and its optimality analysis.

[5]Note that our notion of *virtual queues* is completely different from and unrelated to the notion of *shadow-queues* proposed earlier in [17], [12] and *virtual queues* proposed in [23].

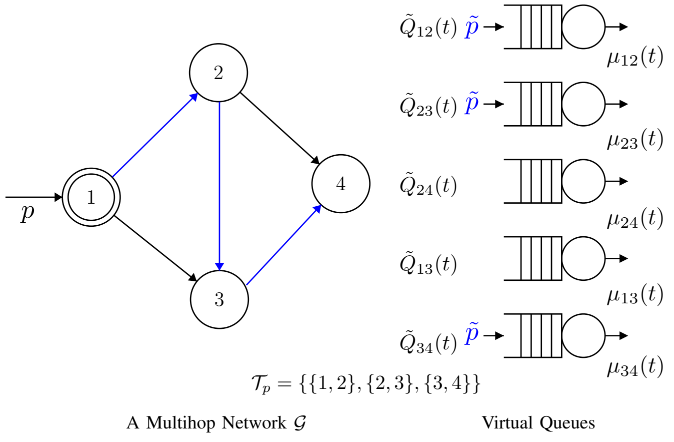
p
1
2
3
4
$\mathcal{T}_p = \{\{1,2\},\{2,3\},\{3,4\}\}$
A Multihop Network $\mathcal{G}$
$\tilde{Q}_{12}(t)$ $\tilde{p}$
$\mu_{12}(t)$
$\tilde{Q}_{23}(t)$ $\tilde{p}$
$\mu_{23}(t)$
$\tilde{Q}_{24}(t)$
$\mu_{24}(t)$
$\tilde{Q}_{13}(t)$
$\mu_{13}(t)$
$\tilde{Q}_{34}(t)$ $\tilde{p}$
$\mu_{34}(t)$
Virtual Queues

Fig. 1: Illustration of the virtual queue system for the four-node network $\mathcal{G}$. Upon arrival, the incoming packet p, belonging to a unicast session from node 1 to 4, is prescribed a path $\mathcal{T}_p = \{\{1,2\},\{2,3\},\{3,4\}\}$. Relaxing the precedence constraints, the packet p is counted as an arrival to the virtual queues $\tilde{Q}_{12}$ and $\tilde{Q}_{23}$ and $\tilde{Q}_{34}$ simultaneously at the *same slot*. In the physical system, the packet p may take a while before reaching any edge in its path depending on the control policy.

B. The Virtual Queue Process $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 1}$

The *Virtual queue process* $\tilde{\boldsymbol{Q}}(t) = \big(\tilde{Q}_e(t), e \in E\big)$ is an $|E| = m$ dimensional controlled stochastic process, imitating a fictitious queueing network *without the precedence constraints*. In particular, when a packet p of class c arrives at the source node $s^{(c)}$, a dynamic policy π prescribes a suitable route $T^{(c)}(t) \in \mathcal{T}^{(c)}$ to the packet. Denoting the set of all edges in the route $T^{(c)}(t)$ by $\{l_1, l_2, \ldots, l_k\}$, this incoming packet induces a virtual arrival *simultaneously* at each of the virtual queues $\big(\tilde{Q}_{l_i}\big), i = 1, 2, \ldots, k$, right upon its arrival to the source. Since the virtual network is assumed to be relaxed with no precedence constraints, any packet present in the virtual queue is eligible for service. See Figure 1 for an illustration.
The (controlled) service process allocated to the virtual queues is denoted by $\{\boldsymbol{\mu}^{\pi}(t)\}_{t\geq 1}$. We require the service process to satisfy the same activation constraints as in the original system, i.e., $\boldsymbol{\mu}^{\pi}(t) \in \mathcal{M}, \forall t$.
Let $A_e^{\pi}(t)$ is the total number of virtual packet arrival (from all classes) to the virtual queue $\tilde{Q}_e$ at time t under the action of the policy π, i.e.,

$$A_e^{\pi}(t) = \sum_{c\in\mathcal{C}} A^{(c)}(t)\mathbb{1}\big(e \in T^{(c)}(t)\big), \;\; \forall e \in E. \tag{5}$$

Hence, we have the following one-step evolution (Lindley recursion) of the virtual queue process $\{\tilde{Q}_e(t)\}_{t\geq 1}$:

$$\tilde{Q}_e(t+1) = \big(\tilde{Q}_e(t) + A_e^{\pi}(t) - \mu_e^{\pi}(t)\big)^+, \;\; \forall e \in E, \tag{6}$$

We emphasize that $A_e^{\pi}(t)$ depends on the route $T^{(c)}(t)$ that the policy chooses, from the set of all admissible routes $\mathcal{T}^{(c)}, \forall c$. An optimal choice of $T^{(c)}(t)$ is given as follows.

C. Dynamic Control and Stability of the Virtual Queues

Next, we design a dynamic routing and link activation policy for the virtual network, which stabilizes the virtual queue process $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 1}$, for all arrival rate-vectors $\boldsymbol{\lambda} \in \text{int}(\overline{\boldsymbol{\Lambda}})$. This policy is obtained by minimizing the one-step drift of a quadratic Lyapunov-function of the *virtual queue lengths* (as opposed to the real queue lengths used in the Back-Pressure policy [13]). In the following section, we will show that when this dynamic policy is used in conjunction with a suitable packet scheduling policy in the physical network, the overall policy is throughput-optimal.
To derive a stabilizing policy for the virtual network, consider a quadratic Lyapunov function $L(\tilde{\boldsymbol{Q}}(t))$ defined in terms of the virtual queue lengths:

$$L(\tilde{\boldsymbol{Q}}(t)) = \sum_{e\in E} \tilde{Q}_e^2(t)$$

From the one-step dynamics of the virtual queues (6), we have:

$$\begin{aligned}\tilde{Q}_e(t+1)^2 &\leq (\tilde{Q}_e(t) - \mu_e^\pi(t) + A_e^\pi(t))^2\\ &= \tilde{Q}_e^2(t) + (A_e^\pi(t))^2 + (\mu_e^\pi(t))^2 + 2\tilde{Q}_e(t)A_e^\pi(t)\\ &- 2\tilde{Q}_e(t)\mu_e^\pi(t) - 2\mu_e^\pi(t)A_e^\pi(t)\end{aligned}$$

Since $\mu_e^\pi(t) \geq 0$ and $A_e^\pi(t) \geq 0$, we have

$$\begin{aligned}\tilde{Q}_e^2(t+1) - \tilde{Q}_e^2(t) &\leq (A_e^\pi(t))^2 + (\mu_e^\pi(t))^2\\ &+ 2\tilde{Q}_e(t)A_e^\pi(t) - 2\tilde{Q}_e(t)\mu_e^\pi(t)\end{aligned}$$

Hence, the one-step Lyapunov drift $\Delta^\pi(t)$, conditional on the current virtual queue lengths $\tilde{\boldsymbol{Q}}(t)$, under the operation of any admissible Markovian policy $\pi \in \Pi$ is upper-bounded by

$$\begin{aligned}\Delta^\pi(t) &\overset{\text{def}}{=} \mathbb{E}\big(L(\tilde{\boldsymbol{Q}}(t+1)) - L(\tilde{\boldsymbol{Q}}(t))|\tilde{\boldsymbol{Q}}(t)\big)\\ &\leq B + \underbrace{2\sum_{e\in E}\tilde{Q}_e(t)\mathbb{E}\big(A_e^\pi(t)|\tilde{\boldsymbol{Q}}(t)\big)}_{(a)}\\ &- \underbrace{2\sum_{e\in E}\tilde{Q}_e(t)\mathbb{E}\big(\mu_e^\pi(t)|\tilde{\boldsymbol{Q}}(t)\big)}_{(b)} \qquad (7)\end{aligned}$$

where B is a constant, bounded by $\sum_e(\mathbb{E}(A_e^\pi(t))^2 + \mathbb{E}(\mu_e^\pi(t))^2) \leq A_{\max}^2 + m$.
The upper-bound on the drift, given by (7), holds good for any admissible policy in the virtual network. In particular, by minimizing the upper-bound point wise, and exploiting the separable nature of the objective, we derive the following decoupled dynamic routing and link activation policy for the virtual network:

Dynamic Routing Policy: The optimal route for each class c, over the set of all admissible routes, is selected by minimizing the following cost function, identified by (a) in Eqn. (7)

$$\mathsf{RoutingCost}^\pi \equiv \sum_{e\in E}\tilde{Q}_e(t)A_e^\pi(t).$$

We remind the reader that $A_e^\pi(t)$ is the routing-policy-induced arrival to the virtual queue corresponding to the link e at time t.
Using Eqn. (5), we may rewrite the objective-function as

$$\mathsf{RoutingCost}^\pi = \sum_{c\in\mathcal{C}} A^{(c)}(t)\Big(\sum_{e\in E}\tilde{Q}_e(t)\mathbb{1}\big(e\in T^{(c)}(t)\big)\Big) \qquad (8)$$

Using the separability of the objective (8) across classes, the above optimization problem decomposes into the following min-cost route-selection problem $T_{\text{opt}}^{(c)}(t)$ for each class c:

$$T_{\text{opt}}^{(c)}(t) \in \arg\min_{T^{(c)}\in\mathcal{T}^{(c)}}\Big(\sum_{e\in E}\tilde{Q}_e(t)\mathbb{1}\big(e\in T^{(c)}\big)\Big) \qquad (9)$$

Depending on the type of flow of class c, the optimal route-selection problem (9) reduces to one of the following well-known combinatorial problems on the weighted graph $\mathcal{G}$, with its edges weighted by the current virtual queue lengths $\tilde{\boldsymbol{Q}}(t)$:

- **Unicast Traffic:** $T_{\text{opt}}^{(c)}(t)$ = The shortest $s^{(c)} - t^{(c)}$ path in the weighted-graph $\mathcal{G}$.
- **Broadcast Traffic:** $T_{\text{opt}}^{(c)}(t)$ = The minimum weight spanning tree rooted at the source $s^{(c)}$, in the weighted-graph $\mathcal{G}$.
- **Multicast Traffic:** $T_{\text{opt}}^{(c)}(t)$ = The minimum weight Steiner tree rooted at the source $s^{(c)}$ and spanning the destinations $\mathcal{D}^{(c)} = \{t_1^{(c)}, t_2^{(c)}, \ldots, t_k^{(c)}\}$, in the weighted-graph $\mathcal{G}$.
- **Anycast Traffic:** $T_{\text{opt}}^{(c)}(t)$ = The shortest of the k shortest $s^{(c)} - t_i^{(c)}$ paths, $i = 1, 2, \ldots, k$, in the weighted-graph $\mathcal{G}$.

Thus, the routes are selected according to a *dynamic source routing* policy [18]. Apart from the minimum weight Steiner tree problem for the multicast traffic (which is NP-hard with several known efficient approximation algorithms [25]), all of the above routing problems on the *weighted* virtual graph may be solved efficiently using standard algorithms [26].

Dynamic Link Activation Policy: A feasible link activation schedule $\boldsymbol{\mu}^*(t) \in \mathcal{M}$ is dynamically chosen at each slot by *maximizing* the term identified by (b) in Eqn. (7), given as follows:

$$\boldsymbol{\mu}^*(t) \in \arg\max_{\boldsymbol{\mu}\in\mathcal{M}}\Big(\sum_{e\in E}\tilde{Q}_e(t)\mu_e\Big) \qquad (10)$$

This is the well-known max-weight scheduling policy, which can be solved efficiently under various interference models (e.g., *Primary* or node-exclusive model [27]).
In solving the above routing and scheduling problems, we tacitly made the assumption that the virtual queue vector $\tilde{\boldsymbol{Q}}(t)$ is *globally known* at each slot, which might not be practically feasible. We will discuss practical distributed implementation of our algorithm in section VI.
In the following Theorem, we establish stability of the virtual queues under the above policy, which will be instrumental for

proving throughput-optimality of the **UMW** policy.

Theorem 2. *Under the above dynamic routing and link scheduling policy, the virtual queue process $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 0}$ is strongly stable for any arrival rate $\boldsymbol{\lambda} \in \text{int}(\bar{\boldsymbol{\Lambda}})$, i.e.,*

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{e\in E}\mathbb{E}(\tilde{Q}_e(t)) < \infty.$$

Proof. The proof involves a stochastic Lyapunov drift argument. See Appendix B of [19] for details. □

As a consequence of the strong stability of the virtual queues $\{\tilde{Q}_e(t), e \in E\}$, we have the following sample-path result, which will be the key to our subsequent analysis:

Lemma 1. *Under the action of the above policy, we have for any $\boldsymbol{\lambda} \in \text{int}(\bar{\boldsymbol{\Lambda}})$:*

$$\lim_{t\to\infty}\frac{\tilde{Q}_e(t)}{t} = 0, \ \ \forall e \in E, \quad \text{w.p. } 1.$$

In other words, the virtual queues are rate-stable [28].

Proof. See Appendix D of [19]. □

The sample path result of Lemma 1 may be interpreted as follows: For any given realization ω of the underlying sample space Ω, define the function

$$F(\omega, t) = \max_{e\in E}\tilde{Q}_e(\omega, t).$$

Note that, for any $t \in \mathbb{Z}_+$, due to the boundedness of arrivals per slot, the function $F(\omega, t)$ is well-defined and finite. In view of this, Lemma (1) states that under the action of the **UMW** policy, $F(\omega, t) = o(t)$ *almost surely*. [6] This result will be used in our sample pathwise stability analysis of the physical queueing process $\{\boldsymbol{Q}(t)\}_{t\geq 0}$.

D. Consequence of the Stability of the Virtual Queues

It is apparent from the virtual queue evolution equation (6), that the stability of the virtual queues under the **UMW** policy implies that the arrival rate at each virtual queue is *at most* the service rate offered to it under the **UMW** routing and scheduling policy. In other words, *effective load* of each edge e in the virtual system is at most unity. This is a necessary condition for stability of the physical queues when the same routing and link activation policy is used for the multi-hop physical network. In the following, we make the notion of "effective-load" mathematically precise.

[6] $g(t) = o(t)$ if $\lim_{t\to\infty}\frac{g(t)}{t} = 0$.

Skorokhod Mapping: Iterating on the system equation (6), we obtain the following well-known discrete time Skorokhod-Map representation [29] of the virtual queue dynamics

$$\tilde{Q}_e(t) = \bigg(\sup_{1\leq\tau\leq t}\big(A_e^\pi(t-\tau,t) - S_e^\pi(t-\tau,t)\big)\bigg)^+, \qquad (11)$$

where $A_e^\pi(t_1,t_2) \stackrel{\text{def}}{=} \sum_{\tau=t_1}^{t_2-1}A_e^\pi(\tau)$, is the total number of arrivals to the virtual queue $\tilde{Q}_e$ in the time interval $[t_1, t_2)$ and $S_e^\pi(t_1,t_2) \stackrel{\text{def}}{=} \sum_{\tau=t_1}^{t_2-1}\mu_e^\pi(\tau)$, is the total amount of service allocated to the virtual queue $\tilde{Q}_e$ in the interval $[t_1, t_2)$. For completeness, we provide a proof of Eqn. (11) in Appendix C of [19].

Combining Equation (11) with Lemma 1, we conclude that under the **UMW** policy, *almost surely* for any sample path $\omega \in \Omega$, for each edge $e \in E$ and any $t_0 < t$, we have

$$A_e(\omega; t_0, t) \leq S_e(\omega; t_0, t) + F(\omega, t), \qquad (12)$$

where $F(\omega, t) = o(t)$.

Implications for the Physical Network: Note that, every packet arrival to a virtual queue $\tilde{Q}_e$ at time t corresponds to a packet in the physical network, that will *eventually* cross the edge e. Hence the loading condition (12) implies that under the **UMW** policy, the total number of packets injected during any time interval $(t_0, t]$, willing to cross the edge e, is less than the total amount of service allocated to the edge e in that time interval up to an additive term of $o(t)$. Thus informally, the "effective load" of any edge $e \in E$ is at most unity.

By utilizing the sample-path result in Eqn. (12), in the following section we show that there exists a simple packet scheduling scheme for the physical network, which guarantees the stability of the physical queues, and consequently, throughput-optimality.

V. Optimal Control of the Physical Network

With the help of the virtual queue structure as defined above, we next focus our attention on designing a throughput-optimal control policy for the multi-hop physical network. As discussed in Section II, a control policy for the physical network consists of three components, namely (1) Routing, (2) Link-activations and (3) Packet-scheduling. In the proposed **UMW** policy, the (1) Routing and (2) Link-activations for the physical network is done exactly in the same way as in the virtual network, based on the current values of the virtual queue state-variables $\tilde{\boldsymbol{Q}}(t)$, described in Section IV-C. There exist many possibilities for the third component, namely the packet scheduler, which efficiently resolves contention when multiple packets attempt to cross an active edge e at the same time-slot t. Popular choices for the packet scheduler include FIFO, LIFO etc. In this paper, for technical reasons, we focus on a particular scheduling policy which has its origin in the context of *adversarial queueing theory* [30]. In particular, we extend the *Nearest To Origin* (NTO) policy to the generalized network flow setting in wireless network, where a packet is allowed to be duplicated. This policy was proposed in [31] in the context of wired networks for the unicast problem. Our

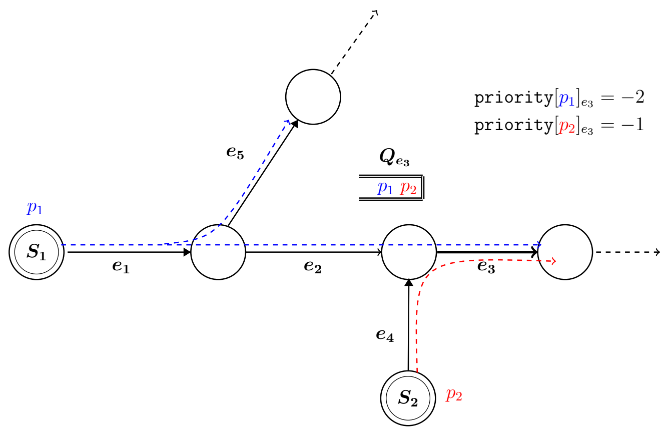

Fig. 2: A schematic diagram showing the scheduling policy ENTO in action. The packets p_1 and p_2 originate from the sources S_1 and S_2. Part of their assigned routes are shown in blue and red respectively. The packets contend for crossing the active edge e_3 at the same time slot. According to the ENTO policy, the packet p_2 has higher priority (having crossed a single edge e_4 from its source) than p_1 (having crossed two edges e_1 and e_2 from its source) for crossing the edge e_3. Note that, although a copy of p_1 might have already crossed the edge e_5, this edge does not fall in the path connecting the source S_1 to the edge e_3 and hence does not enter into priority calculations.

proposed scheduling policy is called Extended NTO (**ENTO**) and is defined as follows:

Definition 2 (Extended NTO)**.** *If multiple packets attempt to cross an active edge e at the same time slot t, the Extended Nearest To Origin (**ENTO**) policy gives priority to the packet which has traversed the least number of hops along its path from its origin up to the edge e.*

The Extended NTO policy may be easily implemented by maintaining a single priority queue [26] per edge. The initial priority of each incoming packet at the source is set to zero. Upon transmission by any edge, the priority of a transmitted packet is decreased by one. The transmitted packet is then copied into the next-hop priority queue(s) (if any) according to its assigned route. See Figure 2 for an illustration. The pseudocode for the full **UMW** policy is given in Algorithm 1.

We next state the following theorem which proves stability of the physical queues under the **ENTO** policy:

Theorem 3. *Under the action of the **UMW** policy with **ENTO** packet scheduling, the physical queues are rate-stable [28] for any arrival vector $\boldsymbol{\lambda} \in \text{int}(\boldsymbol{\Lambda})$, i.e.,*

$$\lim_{t\to\infty} \frac{\sum_{e\in E} Q_e(t)}{t} = 0, \quad \text{w.p.}\,1$$

Algorithm 1 Universal Max-Weight Algorithm (**UMW**) at slot t for the Generalized Flow Problem in a Wireless Network

```
Require: Graph $\mathcal{G}(V,E)$, Virtual queue lengths $\{\tilde{Q}_e(t), e \in E\}$ at the slot $t$.
1: [Edge-Weight Assignment] Assign each edge of the graph $e \in E$ a weight $W_e(t)$ equal to $\tilde{Q}_e(t)$, i.e.
       $\boldsymbol{W}(t) \leftarrow \tilde{\boldsymbol{Q}}(t)$
2: [Route Assignment] Compute a Minimum Weight Route $T^{(c)}(t) \in \mathcal{T}^{(c)}(t)$ for a class $c$ incoming packet in the weighted graph $\mathcal{G}(V,E)$, according to Eqn. (9).
3: [Link Activations] Choose the activation $\boldsymbol{\mu}(t)$ from the set of all feasible activations $\mathcal{M}$, which maximizes the total activated link-weights, i.e.
       $\boldsymbol{\mu}(t) \leftarrow \arg\max_{\boldsymbol{s}\in\mathcal{M}} \boldsymbol{s}\cdot\boldsymbol{W}(t)$
4: [Packet Forwarding] Forward physical packets from the physical queues over the activated links according to the ENTO scheduling policy.
5: [Virtual Queue Counter Update] Update the virtual queues assuming a precedence-relaxed system, i.e.,
       $\tilde{Q}_e(t+1) \leftarrow \Big(\tilde{Q}_e(t) + A_e(t) - \mu_e(t)\Big)^+, \quad \forall e \in E$
```

Proof. This theorem is proved by extending the argument of Gamarnik [31] and combining it with the sample path loading condition in Eqn. (12). See Appendix E of [19] for details. □

As a direct consequence of Theorem 3, we have the main result of this paper:

Theorem 4. *The **UMW** policy is throughput-optimal.*

Proof. For any class $c \in \mathcal{C}$, the number of packets $R^{(c)}(t)$, received by all nodes $i \in \mathcal{D}^{(c)}$ may be bounded as follows:

$$A^{(c)}(0,t) - \sum_{e\in E} Q_e(t) \overset{(*)}{\leq} R^{(c)}(t) \leq A^{(c)}(0,t), \tag{13}$$

where the lower-bound $(*)$ follows from the observation that if a packet p of class c has not reached at all destination nodes $\mathcal{D}^{(c)}$, then at least one copy of it must be present in some physical queue.
Dividing both sides of Eqn. (13) by t, taking limits and using SLLN and Theorem 3, we conclude that w.p. 1

$$\lim_{t\to\infty} \frac{R^{(c)}(t)}{t} = \lambda^{(c)}$$

Hence from the definition (1), we conclude that **UMW** is throughput-optimal.

□

VI. Distributed Implementation

The **UMW** policy in its original form, as given in Algorithm 1, is centralized in nature. This is because the sources need to know the topology of the network and the current value of the virtual queues $\tilde{\boldsymbol{Q}}(t)$ to solve the shortest route and the Max-Weight problems at steps (2) and (3) of the algorithm. Although the topology of the network may be obtained efficiently by topology discovery algorithms [32], keeping track of the virtual queue evolution (Eqn. (6)) is more subtle. Note that, in the special case where all packets arrive only at a single source node, no information exchange is necessary and the virtual queue updates (Step 5) may be implemented at the source locally. In the general case with multiple sources, it is necessary to periodically exchange packet arrival information among the sources to implement Step 5 exactly. To circumvent this issue a heuristic version of **UMW** policy, referred to as **UMW** (heuristic) in the following, may be used in practice where physical queue lengths $\boldsymbol{Q}(t)$ are used as a surrogate for the virtual queue lengths $\tilde{\boldsymbol{Q}}(t)$ for weight and cost computations in Algorithm 1. Routing based on physical queue lengths still requires an exchange of queue length information. However, this can be implemented efficiently using the standard distributed Bellman-Ford algorithm. The simulation results in section VII-B show that the heuristic policy works well in practice and its delay performance is substantially better than the virtual queue based optimal **UMW** policy in wireless networks.

VII. Numerical Simulation

A. Delay Improvement Compared to the Back Pressure Policy - the Unicast Setting

To empirically demonstrate superior performance of the **UMW** policy over the Backpressure policy in the unicast setting, we simulate both policies in the wired network shown in Figure 3. All links have unit capacity. We consider two concurrent unicast sessions with source-destination pairs given by $(s_1 = 1, t_1 = 8)$ and $(s_2 = 5, t_2 = 2)$ respectively. It is easy to see that $\textsf{Max-Flow}(s_1 \to t_1) = 2$ and $\textsf{Max-Flow}(s_2 \to t_2) = 1$ and there exist mutually disjoint paths to achieve the optimal rate-pair $(\lambda_1, \lambda_2) = (2, 1)$. Assuming Poisson arrivals at the sources s_1 and s_2 with intensities $\lambda_1 = 2\rho$ and $\lambda_2 = \rho$, $0 \leq \rho \leq 1$, where ρ denotes the "load-factor", Figure 4 shows a plot of total average queue lengths as a function of the load factor ρ under the operation of the **BP**, **UMW** (optimal) and **UMW** (heuristic) policy.

From the plot, we conclude that both the optimal and heuristic **UMW** policies outperforms the **BP** policy in terms of average queue lengths, and hence (by Little's Law), end-to-end delay, especially in low-to-moderate load regime. The primary reason being, the **BP** policy, in principle, explores all possible paths to route packets to their destinations. In the low-load regime, the packets may also cycle in the network indefinitely, which increases their latency. The **UMW** policy, on the other hand, transmits all packets along "optimal" acyclic routes. This results in substantial reduction in end-to-end delay. We also note that the heuristic version outperforms the optimal policy in terms of delay. This is explored further in the following.

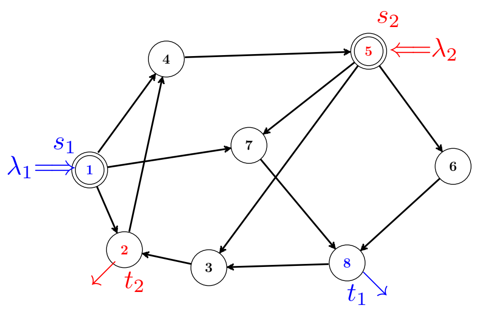

Fig. 3: The wired network topology used for unicast simulation

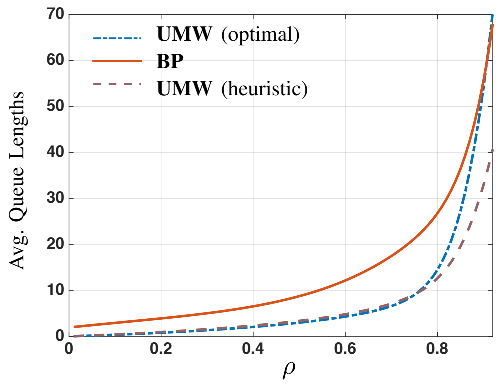

Fig. 4: Comparing the delay-performances of the **BP** and **UMW** (optimal and heuristic) policies in the unicast setting of Fig. 3.

B. Using the Heuristic **UMW** policy for Improved Latency in the Wireless Networks - the Broadcast Setting

Next, we empirically demonstrate that the heuristic **UMW** policy not only achieves the full broadcast capacity but yields better delay performance in this particular wireless network. As discussed earlier, the heuristic policy is practically easier to implement in a distributed fashion. We simulate a 3×3 wireless grid network shown in Figure 5, with *primary* interference constraints [20]. The broadcast capacity of the network is computed to be $\lambda^* = \frac{2}{5}$ [7]. The ENTO policy is used for packet scheduling. The average queue length is plotted in Figure 6 as a function of the packet arrival rate λ under the operation of the (a) **UMW** (optimal) and (b) **UMW** (heuristic) policies. The plot shows that the heuristic policy results in much smaller queue lengths than the optimal policy. The reason being that physical queues capture the network congestion "more accurately" for proper link-activations.

VIII. Conclusion

In this paper we have proposed a new, efficient and throughput-optimal policy, named **U**niversal **M**ax-**W**eight (**UMW**), for the Generalized Network Flow problem. The **UMW** policy can simultaneously handle a mix of Unicast, Broadcast, Multicast and Anycast traffic in arbitrary networks and is empirically shown to exhibit superior performance

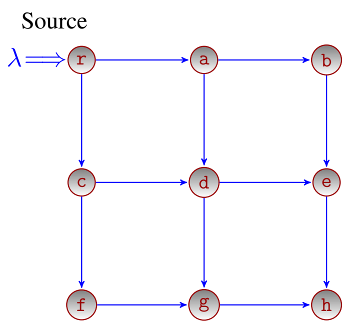

Fig. 5: The wireless topology used for broadcast simulation

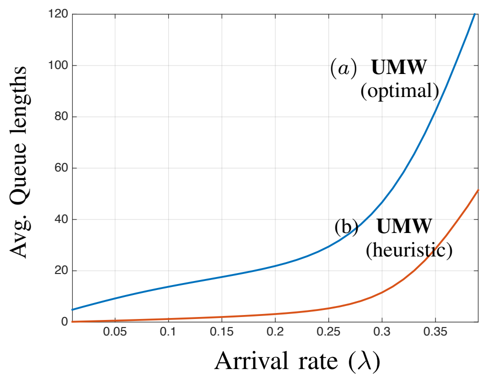

Fig. 6: Comparing Avg. Queue lengths as a function of arrival rate for the optimal (in blue) and the heuristic (in red) **UMW** Policy for the grid network in Figure 5 in the **broadcast** setting.

compared to the existing policies. The next step would be to formally investigate whether the **UMW** policy still retains its throughput-optimality when implemented with physical queue lengths, instead of the virtual queue lengths. An affirmative answer to this question would imply a more efficient implementation of the policy.

References

[1] R. Rustin, *Combinatorial Algorithms*. Algorithmics Press, 1973.

[2] A. Czumaj and W. Rytter, "Broadcasting algorithms in radio networks with unknown topology," in *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*. IEEE, 2003, pp. 492–501.

[3] J. Widmer, C. Fragouli, and J.-Y. Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *In Proceedings*, no. LCA-CONF-2005-016, 2005.

[4] E. Kranakis, D. Krizanc, and A. Pelc, "Fault-tolerant broadcasting in radio networks," *Journal of Algorithms*, vol. 39, no. 1, pp. 47–67, 2001.

[5] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 1073–1081.

[6] D. Towsley and A. Twigg, "Rate-optimal decentralized broadcasting: the wireless case," in *ACITA*, 2008.

[7] A. Sinha, G. Paschos, and E. Modiano, "Throughput-optimal multi-hop broadcast algorithms," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '16. New York, NY, USA: ACM, 2016, pp. 51–60. [Online]. Available: http://doi.acm.org/10.1145/2942358.2942390

[8] A. Sinha, G. Paschos, C.-p. Li, and E. Modiano, "Throughput-optimal broadcast on directed acyclic graphs," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1248–1256.

[9] A. Sinha, L. Tassiulas, and E. Modiano, "Throughput-optimal broadcast in wireless networks with dynamic topology," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '16. New York, NY, USA: ACM, 2016, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/2942358.2942389

[10] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2003.

[11] S. Sarkar and L. Tassiulas, "A framework for routing and congestion control for multicast information flows," *Information Theory, IEEE Transactions on*, vol. 48, no. 10, pp. 2690–2708, 2002.

[12] L. Bui, R. Srikant, and A. Stolyar, "Optimal resource allocation for multicast sessions in multi-hop wireless networks," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1872, pp. 2059–2074, 2008.

[13] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.

[14] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 3, 2011.

[15] M. Zargham, A. Ribeiro, and A. Jadbabaie, "Accelerated backpressure algorithm," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 2269–2275.

[16] A. Sinha, P. Mani, J. Liu, A. Flavel, and D. A. Maltz, "Distributed load management in anycast-based CDNs," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.

[17] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2936–2940.

[18] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*. Springer, 1996, pp. 153–181.

[19] A. Sinha and E. Modiano, "Optimal control for generalized network-flow problems," Tech. Rep., 2016. [Online]. Available: https://arxiv.org/pdf/1611.08641.pdf

[20] C. Joo, X. Lin, and N. B. Shroff, "Greedy maximal matching: Performance limits for arbitrary network graphs under the node-exclusive interference model," *Automatic Control, IEEE Transactions on*, vol. 54, no. 12, pp. 2734–2744, 2009.

[21] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[22] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, 2000.

[23] M. J. Neely, "Energy optimal control for time-varying wireless networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 7, pp. 2915–2934, 2006.

[24] J. K. Lenstra and A. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Operations Research*, vol. 26, no. 1, pp. 22–35, 1978.

[25] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità, "An improved LP-based approximation for Steiner tree," in *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 2010, pp. 583–592.

[26] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[27] L. X. Bui, S. Sanghavi, and R. Srikant, "Distributed link scheduling with constant overhead," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 5, 2009.

[28] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[29] S. Meyn, *Control techniques for complex networks*. Cambridge University Press, 2008.

[30] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and J. Kleinberg, "Universal-stability results and performance bounds for greedy contention-resolution protocols," *Journal of the ACM (JACM)*, vol. 48, no. 1, pp. 39–69, 2001.

[31] D. Gamarnik, "Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM, 1999.

[32] R. Chandra, C. Fetzer, and K. Hogstedt, "A mesh-based robust topology discovery algorithm for hybrid wireless networks," in *Proceedings of AD-HOC Networks and Wireless*, 2002.