Augmenting Backpressure Scheduling and Routing for Wireless Computing Networks

KM Mahfujul, *Student Member, IEEE*, Kaige Qu, *Member, IEEE*, Qiang (John) Ye, *Senior Member, IEEE*, and Ning Lu, *Member, IEEE*

***Abstract*—Driven by the ever-increasing computing capabilities of mobile devices, the next-generation wireless networks are evolving towards distributed networking and computing platforms, which enable in-network computing and unified resource/service provisioning. The evolution leads to a growing research interest in wireless computing networks that operate under the high dynamics of the wireless environment, the complexity of heterogeneous resource allocation, scheduling, and overall optimization. In this paper, we propose a low-complexity efficient solution to jointly allocate both networking resources (e.g., links to forward packets between connected computing nodes) and computing resources (e.g., computing power at each node for packet processing) for wireless computing networks. Specifically, we propose a novel network utility maximization problem under computing and networking resource constraints and develop an enhanced backpressure-based dynamic scheduling and routing algorithm. We verify the network stability and near-optimal performance of the algorithm via both theoretical analysis and extensive simulations.**

***Index Terms*—Wireless computing networks, network utility maximization, cross-layer design, Lyapunov optimization.**

I. Introduction

In evolving towards the fifth generation (5G) and beyond, wireless networks have been experiencing a paradigm shift from a pure communication network to a distributed computing platform, representing a convergence of wireless networking and computing. This is mainly due to the ever-increasing computing power of clients and devices, the emergence of machine-learning-centric applications, and the wide utilization of virtualization and service-oriented principles in various emerging edge/cloud technologies [1], [2]. Such a convergence leads to growing research interests in *wireless computing networks*, which allow in-network computation and integrated resource/function management to realize a unified provisioning of network and computing services. Wireless computing networks are envisaged as the key enablers for emerging applications such as event detection, real-time control and decision-making, streaming data analysis, and many machine-learning-centric applications that require both fast local processing/storage and low-latency networking. A wireless computing network can be abstracted as an undirected graph with a set of computing nodes being vertices and a set of links being edges, where each computing node with certain storage and computing capacities hosts a number of service functions for data processing (such as network diminishers for compressing traffic [3] and machine learning models to process inference requests [4], [5]) and each link with certain communication capacity delivers data traffic over a wireless channel. The wireless computing network is to serve a set of augmented information (AgI) flows [6]. For each AgI flow, the destination node is provided with augmented information generated during the real-time processing of source data packets through a chain of multiple service functions. The service functions are hosted by computing nodes on a respective routing path, for the AgI flow (e.g., Fig. 1). For the rest of the paper, we use the terms service functions and functions interchangeably. Existing works have investigated how to jointly schedule the communication and computation resources, i.e., how to determine where to execute each function and how to optimally route the flows to meet service demands. The control policies for function placements and flow routing has been investigated in either a static or dynamic setting. For example, some studies [7]–[10] consider the static function placement at computing nodes according to the computing/communication requirements and flow routing among the computing nodes. Since the function placement

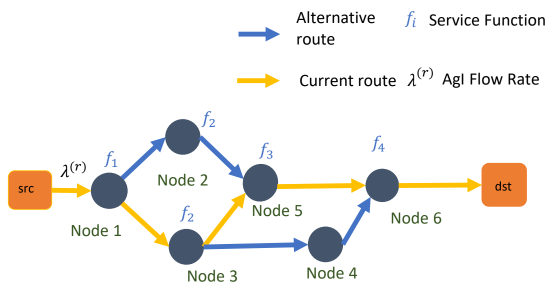


Fig. 1: Example of a wireless computing network with computing nodes and communication links serving an AgI flow r from a source node to a destination node with a flow (packet arrival) rate $\lambda^{(r)}$. The computing network places functions f_i at different computing nodes. Based on the function placement choice at wireless computing nodes, the flow r packets take alternative routing paths towards the destination node.

Manuscript received 12 December 2023; revised 10 May 2024 and 9 August 2024; accepted 06 September 2024. date of current version 11 September 2024. Part of this paper was presented in IEEE ICC 2023.

KM Mahfujul (Corresponding author) and Ning Lu are with the Department of Electrical and Computer Engineering, Queen's University, ON, Canada (email:{m.kadir, ning.lu}@queensu.ca). Kaige Qu is with the Department of Electrical and Computer Engineering University of Waterloo, Waterloo, ON, Canada (email: kaige.qu@uwaterloo.ca). Qiang (John) Ye is with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: qiang.ye@ucalgary.ca).

and flow routing are static and computing nodes are resource-limited, the packet arrival rates for the flows should be throttled (e.g, by packet dropping) in resource shortage. On the other hand, dynamic placement of functions and flow rerouting strategies are investigated in [6], [11], [12] in the presence of resource shortage. Although the flow rerouting process can redistribute the computing loads across the network, it incurs a non-negligible overhead, (e.g., delay/cost of link creation and packet re-transmission [12], [13]). The reconfiguration overhead also affects the scheduling and routing of other flow packets. Therefore, we need a proper trade-off between the computing-communication resource allocation among all the flows, for example, through the network utility maximization (NUM) formulation.

However, unlike traditional wireless networks that route and forward the packets from source to destination, wireless computing networks combine packet processing/computing and routing/forwarding for various service demands. Furthermore, to meet severe changes in service demands, it also necessitates studying control policies that can reconfigure the service flows in resource shortage, for example, by exploring multi-path routing [14], [15]. Due to such unique characteristics of wireless computing networks, formulating an NUM unifying computing and networking is challenging; thus still underexplored in existing research.

In this paper, we consider an NUM problem in a wireless computing network for delivering multiple AgI flows. We consider a wireless computing network with a set of generic computing nodes to compute service functions via virtualization and wireless communication links to route the AgI flows among computing nodes. We assume, the packets from an AgI flow require computing by a pre-defined sequence of computing functions before being delivered to the destination and a computing node hosts all such functions. Each function in a computing node computes the payload of a packet and promotes the packet to the next function in the sequence. Specifically, a function promotes/updates the ***packet state*** defined as follows. After a function completes computing, the function revises the packet header/payload. Furthermore, the computing node forwards the packet to the subsequent function in the same computing node. We denote this whole process (promoting the *packet state* and forwarding to subsequent function in the same node) as ***internal packet forwarding***. Alternatively, a computing node decides to forward the packet to a neighboring computing node using a wireless communication link for computation. We denote this process as ***external packet forwarding***. Note that, the *internal packet forwarding* involves packet promotion and computing resource consumption while *external packet forwarding* involves communication resource consumption. In this paper, we study joint internal-external forwarding of AgI flow packets that result in augmented scheduling and routing in wireless computing networks.

The computing nodes backlog the packets based on their current *state* and operate these packet-forwarding strategies independently across the network with limited information exchange among neighboring computing nodes. The joint packet forwarding uses the resource scheduling in *medium-access control (MAC)-layer* in conjunction with queue backlog information of neighboring computing nodes in *network-layer* for packet routing (i.e., cross-layer control) to offer a low-complexity distributed control policy for a joint computing-communication resource allocation problem. Moreover, the *state*-dependent internal/external packet forwarding scheme explores multi-path routing of flow packets. We formulate the network utility as a function of the flow packet admission rate in the source node to control the network congestion and jointly utilize the *state*-dependent packet forwarding schemes for resource allocation. By utilizing duality, we develop an augmented backpressure-based distributed scheduling and routing algorithm. We summarize the main contributions of this paper in the following.

- We introduce an NUM formulation for AgI flow delivery in wireless computing networks as a convex optimization problem. Different from existing NUM formulations in wireless networks, AgI flows require different computing and communication resource allocations in an end-to-end manner. Due to the additional computing resource constraint, an NUM formulation for AgI flow delivery in wireless computing networks is a unique challenge. To the best of our knowledge, investigating an NUM problem for wireless computing networks is still underexplored in the existing research.
- We address the packet specific resource requirements by augmenting packet *state*-based ***internal-forwarding*** (packet computing and promoting) and ***external-forwarding strategy*** (packet routing). Our formulation considers communication/computing resource trade-off and enables resource allocation in the computing network to steer the flow packets towards the destination via different routes. The proposed algorithm develops a *Lagrange dual*-based cross-layer control policy that utilizes queue backlog information to jointly provide congestion control, computing, and packet routing decisions.
- We theoretically verify the proposed algorithm in terms of network stability and near-optimal network utility. Further, we carried out extensive simulations to demonstrate the performance of the proposed algorithm from various aspects including resource fairness, network stability, and utility. The simulation results also cross-verify our theoretical analysis and the effectiveness of the proposed algorithm for wireless computing networks.

The rest of this paper is organized as follows. In Section II, we discuss the background and related works. In Section III, we introduce the network model and the NUM problem formulation. In Section IV, we elaborate on the design of the dynamic backpressure-based scheduling and routing algorithm by decoupling the problem into solvable sub-problems with solutions. We verify the performance of the proposed algorithm by theoretical analysis in Section V, followed by the simulations in Section VI, and we conclude our paper and discuss the future works in Section VII.

II. Background and Related Works

Consider a wireless network by a collection of nodes and communication links that deliver network flows. In this particular setting, we consider a set of network flows that are simply described by some source-destination node pairs (e.g., wireless sensor networks). The source node generates packets and the packets are collaboratively forwarded by some intermediate nodes acting as routers to deliver the packets to the destination node. The intermediate nodes forward the packets by maintaining some data queues for each specific flow packet. Most importantly, these network flows do not establish a-priory routes, meaning that a flow packet is cooperatively forwarded by any arbitrary intermediate computing nodes towards the destination.

Assuming that the rate of incoming flow packets at the source nodes is inelastic, network flow delivery while achieving a fair transmission rate allocation among the flows and maintaining overall network stability is developed by cross-layer control algorithms. Specifically, a decentralized congestion controller situated at the *transport-layer* works in conjunction with a queue backlog-based scheduler at the *MAC-layer*. Further, the flow packets are routed towards the destination nodes via the intermediate network nodes using the communication links at the *network-layer*. The joint mechanism is established by performing a loose coupling among the layers to achieve buffer stability, optimal routing, and fair resource allocation [16], [17].

However, due to the unpredictability of wireless channels, the joint decision-making of flow scheduling, routing, and resource allocations becomes very challenging, therefore complicating the overall network design. In the context of traditional networks, *Lyapunov-drift-plus-penalty (LDP)* is a promising approach to tackle such intricate decision-making problem [18], [19]. For example, [19] extends the LDP approach for a multi-hop, multi-commodity wireless ad-hoc network leading to a backpressure algorithm. The proposed algorithm exploits the broadcasting nature of wireless networks and is shown to be throughput optimal.

A key difference from *wireless computing networks* from the traditional wireless networking design can be seen in the following. Existing wireless networking mainly focuses on efficient information delivery from an information source to the destination (e.g., using cross-layer control algorithms). In contrast, wireless computing networks require to deliver the information flows from source to destination while performing some computing on the flow packets by following a service flow delivery model (e.g., AgI flows). Therefore, the intermediate network nodes or so-called computing nodes are powered with computing and communication resources to satisfy that requirement. Being an essential part of wireless computing networks, the distribution of computing resources to execute the service functions at different computing nodes and communication resources to route the flow packets through the appropriate sequence of service functions are our focus.

Given a fixed flow rate (e.g., packet admission rate), linear programming and heuristic procedures for joint computing-communication resource allocations are studied in [9], [10], [12], [20], [21]. The research works [7], [22] generalize the service delivery problem in cloud networks via joint computing/communication resource allocation. However, in response to the unknown changes in the network environment, such as resource shortage/disruption in cloud computing nodes, some readjustments in computing/communication resources are necessary. The study of dynamic network control policies that can reconfigure the computing/communication resource allocations in response to unpredictable demands is studied by [23], [24]. These works provide a characterization of the cloud network capacity region and design throughput-optimal control policies to jointly allocate computing/communication resources while minimizing the overall network costs.

Compared to the traditional centralized cloud networks, distributed wireless computing networks provide extended flexibility in computing/communication resource allocations leading to a clear advantage in meeting stringent service requirements both in latency and cost, handling mobility, and real-time location awareness [11]. By introducing cloud service models to wireless networks, the work in [6] extends the works in [7], [8], [23] for AgI flow delivery in wireless networks and develops a distributed algorithm that utilizes the queue backlog information. The resultant algorithm makes a joint transmission, computing, and resource allocation decision with minimum average network resource utilization cost. While the proposed algorithms in [6], [23] study the minimum cost resource management and dynamically adjust the scheduling decisions, they overlook the fact that reconfiguring the compute and communication resources results in a non-negligible amount of delay leading to additional costs [13]. Recent works in [25], [26] propose joint computing, communication, and caching in cloud networks. In order to process the flow packets that require storage resources, they pre-store static data objects at different network locations that are to be routed via an augmented layered graph (ALG) to model multi-pipeline-based flow control.

The existing literature reviewed earlier in this section covers a wide range of research works that focus on forwarding the network flow packets in wireless networks by either considering single-path/multi-path packet routing ([14]–[19], [27], [28] and [29]–[31]). While the cross-layer control techniques are effective and well-known for multi-hop wireless network flow control, wireless computing networks demand flow packet processing/computing by a set of functions where each packet requires a different combination of computing/communications resources based on the processing steps. In such a service delivery context, AgI service delivery problems are also well studied in cloud networks by considering a predefined set of resource allocation decisions [7]–[10], [22] or taking a resource reconfiguration strategy [11], [12], [20], [23], [24], [32] in either a centralized service function chaining (SFC) (e.g., [32]) or a distributed general cloud network setting. For example, [33] studies the AgI flow packet routing problem under end-end deadline constraints. Specifically, they consider that each flow packet has its own delivery deadline. However, the proposed cloud network accommodates one type of network flow and does not consider flow packet computing. The distributed cloud network designs are further extended by [6] that develops a service delivery mechanism for wireless computing networks. A number of recent studies investigated joint AgI flow chaining and routing with the objective of maximizing the accepted service requests in wireless networks

TABLE I: The key notations

Notation	Definition
$\mathcal{N}, \mathcal{R}, \mathcal{K}$	Set of computing nodes, AgI flows, and current state of packets.
V	Utility scaling parameter.
$\lambda^{(r)}(t)$	Source rate (Admitted flow rate) of AgI flow $r \in \mathcal{R}$.
$Q_n^{(r,k)}(t)$	Data queue backlog in computing node $n \in \mathcal{N}$, belongs to AgI flow $r \in \mathcal{R}$, currently at state $k \in \mathcal{K}$.
$s_{n,j}^{(r,k)}(t)$	External forwarding rate (forwarding rate of the packet from queue $Q_n^{(r,k)}(t)$ to $Q_j^{(r,k)}(t)$ between neighboring computing nodes $n \to j$.
$a_n^{(r,k)}(t)$	Internal forwarding rate (packet promotion rate of from queue $Q_n^{(r,k)}(t)$ to $Q_n^{(r,k+1)}(t)$.
$\sigma^{(r,k)}$	Packet size of AgI flow $r \in \mathcal{R}$.
$\rho^{(r,k)}$	Computing requirement of data packets (in CPU cycles/bit) of AgI flow r currently at state k.
$z_{n,j}^{(r,k)}(t) \in \{0,1\}$	Indicator variable denoting that node n is communicating via wireless link with node j at slot t.
$x_n^{(r,k)}(t) \in \{0,1\}$	Indicator variable denoting packets of AgI flow r currently at state k are scheduled for computing at node n, at slot t.

[34]–[38]. The aforementioned solutions consider static function placement/packet routing in wireless networks.

Different from the literature, our work distinguishes itself by enhancing the flexibility of joint computing/communication resource allocation to flow packets in multi-hop wireless computing networks while taking into account their packet-specific resource requirements. Specifically, we augment state-based ***internal/external packet forwarding*** to minimize the complicacy in joint resource allocation and also isolate the packet forwarding in the computing nodes without impacting the resource allocations on other flow packets in upstream nodes. Furthermore, we introduce NUM formulation to transform AgI flow delivery problem into a convex optimization problem and present a *Lagrange* dual-based backpressure algorithm to address this. In essence, our contribution lies in the integration of dynamic resource allocation given by *state*-based ***internal/external packet forwarding*** and NUM formulation for AgI flow delivery to apply the convex optimization techniques. This not only streamlines cross-layer control in wireless computing networks but also addresses specific challenges that arise when jointly allocating computing and communication resources to flow packets.

III. System Model and Problem Formulation

A. Network Model

We consider a time-slotted wireless computing network with time slots indexed by $t \in \{1, 2, 3, ...\}$. The wireless computing network consists of computing nodes in set $\mathcal{N}$ that are equipped with wireless communication links in set $\mathcal{L}$ to process AgI flows. For each AgI flow $r \in \mathcal{R}$, packets are admitted/generated at a fixed source node $src(r) \in \mathcal{N}$ and packets leave the network at a fixed destination node $dst(r) \in \mathcal{N}$ after being processed by a set of functions that are hosted in the computing nodes. A computing node $n \in \mathcal{N}$ both processes and forwards the packets from AgI flow $r \in \mathcal{R}$. Computing node n hosts computing functions in set $\mathcal{K}$ to process the flow packets from AgI flow r. We assume the AgI flow packets are processed by a set of functions $K^{(r)} \subseteq \mathcal{K}$ in an ordered sequence that is specific to AgI flow r.

To address the joint computing/communication resource allocation in wireless computing networks, we assume that the computing functions $k \in \mathcal{K}$ do not require storage resources. As a result, the computing functions do not have location dependence, i.e., a computing function $k \in K^{(r)} \subseteq \mathcal{K}$ can be hosted at any computing node n. Therefore, without loss of generality, it is safe to assume that each computing node n hosts all computing functions $K^{(r)} \subseteq \mathcal{K}, \forall r \in \mathcal{R}$. For simplicity, we assume the computing network is loop-free. The rest of the paper uses the terms AgI flow and flow interchangeably by abusing the notation r.

B. Packet States

Packets from an AgI flow r require processing by a sequence of computing functions $k \in K^{(r)} \subseteq \mathcal{K}$ at computing node n before being delivered to the destination $dst(r)$. In our work, we relate *packet state* to the packet processing by a computing function k. The detail is given as follows. Each function k processing the packets from flow r updates the packets by revising the packet header/payload. When a packet finishes processing by a function k, meaning the packet completes a specific type of processing (e.g., encryption), the *packet state* is updated (packet header/payload is revised), and scheduled to be processed by the downstream functions in sequence. Since there are $K^{(r)}$ such functions for each r, we associate $K^{(r)}$ functions with $|K^{(r)}|$ packet states, where $|K^{(r)}|$ denotes cardinality of set $K^{(r)}$. For simplicity, we index the functions and *packet states* analogously by k. The rest of the paper uses the tuple (r, k) to refer to a packet of flow r at state k.

C. Internal/External Forwarding of Packets

We consider that a computing node n hosts all the functions, $\{k : k \in K^{(r)} \subseteq \mathcal{K}, \forall r \in \mathcal{R}\}$. To support such processing, node n maintains $K^{(r)}$ data queues that backlog packets of each flow r, indexed by tuple (r, k) representing function k of flow r. To ease the presentation, we abuse the notation to refer data queue using the tuple (r, k) that backlogs all (r, k) packets. The queue dynamics are detailed in sub-section III-D.

A computing node n hosting $K^{(r)}$ functions either *internally* forwards a packet from queue (r,k) to queue $(r,k+1)$ in the same node or *externally* forwards from queue (r,k) to queue (r,k) at a neighboring node $j \in \mathcal{N}$. ***Internal packet forwarding*** is analogous to computing/processing by a function k, promotion/update of *packet state*, and forwarding the packets to the queue $(r,k+1)$ involving computing resource consumption at node n. [1] On the other hand, the ***external packet forwarding*** does not involve a packet promotion since the node n does not compute but forwards the packet to a node j using an active wireless communication link $(n,j) \in \mathcal{L}$.

It is worth mentioning that, the source node of a flow $src(r)$ generates the flow packets and a queue (r,k) at node $src(r)$ performs both internal/external packet forwarding. On the other hand, destination $dst(r)$ cannot externally forward packets. Data queues in set $\{(r,k) : k<|K^{(r)}|, n = dst(r)\}$ at the destination of a flow only internally forward the packets since the $dst(r)$ must finish required computing (if not already computed by upstream computing nodes) and delivers packets to users. Queues in node n (including the $src(r)$) in set $\{(r,k) : k = K^{(r)} = |K^{(r)}|, \forall n \in \mathcal{N} \setminus \{dst(r)\}\ \}$ only externally forward the packets since $k = |K^{(r)}|$ denotes the last function in the sequence specified by flow r. Therefore, a packet that finishes all the required computing steps is externally forwarded to downstream nodes until it reaches the destination. The data queues in set $\{(r,k) : k = |K^{(r)}|, n = dst(r)\}$ process and deliver the packets (e.g., to end users). The details are given later (sub-section III-D). To simplify, we let $K^{(r)} = |\mathcal{K}|$, $\forall r \in \mathcal{R}$. There are in total $|\mathcal{R} \times \mathcal{K}|$ queues at node n in considering all the flows $r \in \mathcal{R}$. We provide an illustrative example of the system model in Fig. 2.

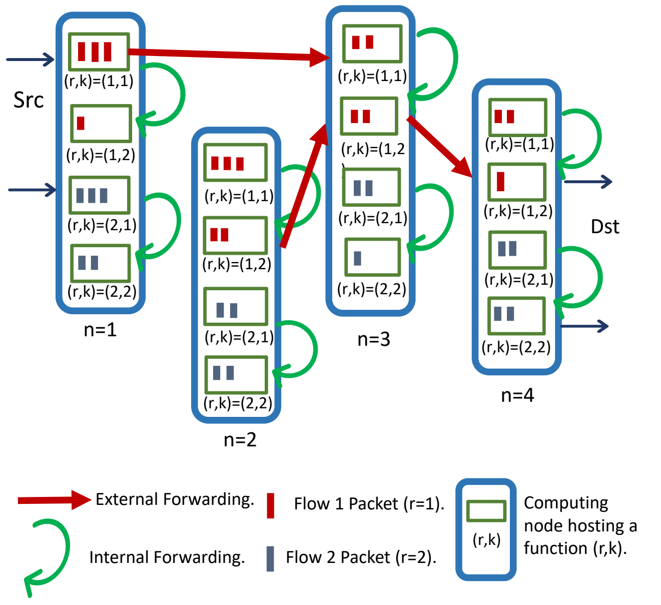


Fig. 2: Wireless Computing Network: A wireless computing network with four nodes, $|\mathcal{R}| = 2$ AgI flows with $|\mathcal{K}| = 2$ states is shown. Node $n = 1$ is the source node (Src), and $n = 4$ is the destination (Dst) of the computing network. Internal forwarding is shown by curved green arrows and external forwarding is shown by straight red arrows. Note that the forwarding arrows do not correspond to a complete flow path but represent some forwarding choices. For example, from $n = 3$ to $n = 4$, there are four candidate queues to externally forward their packets, the algorithm may choose $(r,k) = (1,2)$ to forward its packets at the current slot.

D. Flow Model

Packets from AgI flow $r \in \mathcal{R}$ are admitted into the network through source node $src(r)$. The data packets generated by some users/applications are not admitted directly to the *network layer*. The packets are primarily backlogged at a *transport layer* reservoir and the $src(r)$ node allows packets admission to the *network layer* at a rate $\lambda^{(r)}(t)\mathbb{1}_{(k=1,n=src(r))}$, denoting that the source rate of flow r at the source node $n = src(r)$ and the packet is at its earliest state $k = 1$. [2] The source node $src(r)$ maintains a *network layer* queue $Q^{(r,k=1)}_{n=src(r)}$ that backlogs the packets that are allowed from the *transport layer* at rate $\lambda^{(r)}(t)\mathbb{1}_{(k=1,n=src(r))}$ with the packets that are already backlogged during the previous slots. To simplify the presentation, we abuse the notation to denote queue backlogs by $Q^{(r,k=1)}_{n=src(r)}(t)$ at each time slot. To emphasize the fundamental issues of flow control, routing, and resource allocation, we assume that the *transport layer reservoirs* have infinite backlogs. Such a model with an infinite backlog case is often studied in communication research to evaluate the maximum achievable performance of protocols [16], [29], [39].

Consider source rate is upper-bounded by $\lambda^{(r)}(t)\mathbb{1}_{(k=1,n=src(r))} \leq \lambda^{max}<\infty$. A computing node n processes packets from queue $Q^{(r,k-1)}_n(t)$ (i.e., packets of flow r currently on state $k-1$, in a queue indexed by $(r,k-1)$) at time slot t with a processing rate $a^{(r,k-1)}_n$ and promotes to queue $Q^{(r,k)}_n(t)$ at the same node by the end of slot t. We denote this process as ***internal packet forwarding***. Alternatively, the computing node decides to forward some (r,k) packets to a neighboring node $j \in \mathcal{N}$ at rate $s^{(r,k)}_{n,j}$, denoted as ***external packet forwarding***. The externally forwarded packets are either stored in the queue $Q^{(r,k)}_j(t)$, for processing or further externally forwarded to another computing node in future time slots. Considering all the variables, we present a flow conservation constraint for each (r,k) at slot t by,

$$\lambda^{(r)}(t)\mathbb{1}_{(k=1,n=src(r))} + \sum_{i\in\mathcal{N}} s^{(r,k)}_{i,n}(t) + a^{(r,k-1)}_n(t) \leq \sum_{j\in\mathcal{N}} s^{(r,k)}_{n,j}(t) + a^{(r,k)}_n(t), \quad \forall n \in \mathcal{N}, \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \quad (1)$$

where on the left-hand side, a source node (if only $n = src(r)$) admits packets with state $k = 1$, a node n receives internal packet forwarding from $(r,k-1)$ in the same node at rate

[1]The internal packet forwarding does not involve packet routing using communication links to other computing nodes (i.e., n to j) as in the case of external packet forwarding. Rather, the internal forwarding is operated via an internal link/virtual link from a queue (r,k) to queue $(r,k+1)$ that is internal to a computing node n.

[2]Here, $\mathbb{1}$ is representing an indicator function defined as

$$\mathbb{1}_{(k=1,n=src(r))} = \begin{cases} 1 & if \quad (k = 1 \text{ and } n = src(r)), \\ 0 & \text{otherwise.} \end{cases}$$

$a_n^{(r,k-1)}$, and node $n \in \mathcal{N} \setminus \{src(r)\}$ receives external packet forwarding from the upstream (e.g., from source node) at rate $s_{i,n}^{(r,k)}$. Note that with $k=1$, we consider internal forwarding $a_n^{(r,k-1)}(t)=0$ on the left-hand side. This is a special case where the function $(r,k-1)$ is a dummy function. Similarly, on the right-hand side, the node promotes the packets to the next state (e.g., from $k-1$ to k) at rate $a_n^{(r,k)}$ and internally forwards to the queue $Q_n^{(r,k+1)}(t)$, by the end of slot t. Alternatively, the node externally forwards some packets downstream (e.g., towards the destination) at rate $s_{n,j}^{(r,k)}$. The inequality sign in Eq. (1) is because the total arrival rates should be no more than the total departure rates. We represent the flow conservation constraint via an associated data queue for every (r,k) tuple

$$Q_n^{(r,k)}(t+1) \leq \left[Q_n^{(r,k)}(t) - \sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - a_n^{(r,k)}(t)\right]^+ + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t) + a_n^{(r,k-1)}(t) + \lambda^{(r)}\mathbb{1}_{(k=1,n=src(r))}(t), \quad (2)$$

where the data queues will maintain finite queue backlogs in the network while implementing the resulting algorithm. The computing nodes schedule the internal/external packet forwarding by scheduling the computing/communication resources. The details are given in the following.

1) Computing Resource: Each node has limited CPU processing power (in CPU cycles allocated per time slot) denoted by $\mathcal{C}_n$. Scheduling the queues (indexed by (r,k)) is analogous to performing different types of computing functions. The computing functions require different CPU cycles to process a single bit of flow packets, denoted by $\rho^{(r,k)}$ for flow r at state k. Additionally, we denote the flow packet size by $\sigma^{(r,k)}$ in MB, which are different for all flows $r \in \mathcal{R}$. Computing node n schedules queue $Q_n^{(r,k)}$ (representing a tuple (r,k)) to compute the packets at every time slot, i.e.,

$$x_n^{(r,k)}(t)\cdot a_n^{(r,k)}(t)\cdot\sigma^{(r,k)}\rho^{(r,k)} \leq \mathcal{C}_n,\ \forall(r,k)\in(\mathcal{R},\mathcal{K}), \quad (3)$$

where $x_n^{(r,k)}(t)\in\{0,1\}$ represents scheduling of a particular (r,k) tuple. Eq. (3) represents computing resource constraints at node n. We assume, node n schedules one (r,k) tuple at a time slot (e.g., hardware limitation) i.e.,

$$\sum_{(r,k)\in(\mathcal{R},\mathcal{K})} x_n^{(r,k)}(t) \leq 1, \forall n\in\mathcal{N}. \quad (4)$$

2) Communication Resource: The computing nodes externally forward the flow packets downstream via the communication links $\{(n,j)\in\mathcal{L},\ \text{and}\ n,j\in\mathcal{N}\}$ between nodes using the orthogonal frequency bands. We consider a link routing constraint by which a node n forwards packets to at most one downstream node j at a time slot i.e.,

$$\sum_{j\in\mathcal{N}}\sum_{(r,k)\in(\mathcal{R},\mathcal{K})} z_{n,j}^{(r,k)}(t) \leq 1, \quad \forall n\in\mathcal{N}, \quad (5)$$

where $z_{n,j}^{(r,k)}(t)\in\{0,1\}$ represents scheduling of a (r,k) tuple by using an active wireless communication link (n,j). If there is an active link (n,j) between the nodes n and j, the computing node n is able to send packets using that link. We denote, the allowable link-transmission rate by a vector $\boldsymbol{\mu}_n(t)=\{\mu_{n,j}(t),..\}$, where, $(n,j)\in\mathcal{L}$, with a finite upper-bound on the rate as $\mu_{n,j}^{max}<\infty$. As a reasonable assumption, we define $\tilde{\boldsymbol{\Upsilon}}\in\mathbb{R}_+^{|\mathcal{L}|}$, as a bounded region, representing the achievable rate $\boldsymbol{\mu}_n$ under the link constraint (5) at a time slot. Without loss of generality, the network allows a discrete set of achievable rates, thus the set is non-convex. The achievable link rates are the result of the power assignment on the corresponding links. The network manages a timesharing procedure among the different available rates $\tilde{\boldsymbol{\Upsilon}}$ by simply defining a convex hull, i.e., $\boldsymbol{\Upsilon} := \mathcal{CO}\{\tilde{\boldsymbol{\Upsilon}}\}$, that provides a closed and bounded capacity region.

Definition 1. *(Link-rate Capacity Region): The capacity region, Λ is a set so that we can capture a set of potentially achievable rates (the maximum allowable rate facilitated by the link) satisfying the following constraint, $[\sum_{(r,k)}\{\mu_{n,j}^{(r,k)}(t)\}]\in\boldsymbol{\Upsilon}$, where $\mu_{n,j}^{(r,k)}(t)\geq 0,\forall(n,j)\in\mathcal{L}, r\in\mathcal{R}$. Further, due to the constraint (5), we choose one (r,k) tuple to send over the link. Thus, we revise the capacity constraint $[\sum_{r,k}\{\mu_{n,j}^{(r,k)}(t)\}]\in\boldsymbol{\Upsilon}$ from above to a new simpler constraint as, $[\{\mu_{n,j}(t)\}]\in\boldsymbol{\Upsilon},\forall(n,j)\in\mathcal{L}$.*

Remark 1. *Due to the shared nature of wireless communication media, the data rate on link (n,j) does not only depend on the power assignment on the link (n,j) but also the interference resulting from the power assignment on the other interfering links. Thus the link scheduling problem is assumed to be centralized. We discuss an alternative approach in **Appendix G**.*

The packets are externally forwarded over the link (n,j), to the downstream at a rate $s_{n,j}^{(r,k)}(t)$, denoting the actual packet forwarding rate. We provide the relationship with potential packet forwarding rate (the maximum allowable rate facilitated by the link) $\mu_{n,j}(t)$ by,

$$s_{n,j}^{(r,k)}(t)\cdot\sigma^{(r,k)} \leq z_{n,j}^{(r,k)}(t)\cdot\mu_{n,j}(t),\ \forall(r,k)\in(\mathcal{R},\mathcal{K}). \quad (6)$$

Suppose the wireless computing network operations can be described by the Eq. (1)-(6) for all flows $r\in\mathcal{R}$, all computing nodes $n\in\mathcal{N}$, and the available wireless communication links between two neighboring computing nodes $(n,j)\in\mathcal{L}$. We also assume that under such a wireless computing network, a dynamic algorithm yields $\lambda^{(r)}(t), a_n^{(r,k)}(t)$, and $s_{n,j}^{(r,k)}(t)$ rates at each time slot t, and all queues are initially empty (i.e., $Q_n^{(r,k)}(0)=0$).

E. Problem Formulation

Assumption 1. *We established an instantaneous source rate $\lambda^{(r)}(t)$ for each AgI flow r. We assume that this source rate has a well-defined time average rate given by,*

$$\overline{\lambda}^{(r)}(T) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{\lambda^{(r)}(t)\}.$$

To maximize the total source rates of all flows $r\in\mathcal{R}$, we use a network utility function for each flow r as a concave increasing function of time average source rate denoted by

$U_r(\overline{\lambda}^{(r)}(T))$. We define $U_r(\cdot)$ as a concave [3], non-decreasing function of flow rate which is continuously differentiable with respect to $\lambda^{(r)}$ over a bounded domain. Such a utility function is often advocated in the context of communication networks [16], [39], [40]. Considering all the network constraints, we formally present the NUM problem:

$$\textbf{(P1)} \max_{\substack{\lambda^{(r)}(t), a_n^{(r,k)}(t), s_{m,n}^{(r,k)}(t),\\ z_{n,j}^{(r,k)}(t), x_n^{(r,k)}(t)}} \sum_{r=1}^{R} U_r(\overline{\lambda}^{(r)}(T)) \tag{7}$$

$$\text{subject to} \quad (1), (3), (4), (5), (6),$$
$$\forall n \in \mathcal{N}, \quad \forall r \in \mathcal{R}, \quad \forall k \in \mathcal{K}, \quad \forall t.$$

Here, it is important to note that the objective of problem **(P1)** is network utility maximization in terms of source rate subject to the underlying computing/communication resource allocation of flow packets at each computing node n. The joint computing/communication resource allocation is managed by augmenting an internal/external packet forwarding mechanism that allocates computing/communication resources to queues indexed by (r, k) representing backlog at a computing function. We assume, the packet admission of each flow r at $src(r)$, computing resource allocation (e.g., which (r, k) queue internally forwards packets), communication resource (e.g., which (r, k) queue externally forwards packets to downstream node) is yielded by the dynamic algorithm at each time slot (see **Section IV**). One important aspect of this approach is, rather than a pre-determined resource allocation to AgI flow packets, the algorithm dynamically allocates computing resources to internally forward or communication resources to externally forward the packets.

IV. Dynamic Backpressure-based Framework

First, we define a vector to capture all the decision variables of the optimization problem **(P1)**,

$$y_n^{(r,k)}(t) = \begin{cases} [\lambda^{(r)}(t)\mathbb{1}_{(k=1, n=src(r))}, s_{n,j}^{(r,k)}(t), a_n^{(r,k)}(t)], \\ \qquad n = src(r), j \in \mathcal{N}, k = [1, \mathcal{K}], \\ \text{and} \\ [s_{n,j}^{(r,k)}(t), a_n^{(r,k)}(t)], \\ \qquad n \neq src(r), j \in \mathcal{N}, k = [1, \mathcal{K}]. \end{cases}$$

The first case summarizes all the decision variables when the computing node n is the source node, therefore we include source rate $\lambda^{(r)}(t)$ and also the queue can externally forward packets to some other neighborhood node j with rate $s_{n,j}^{(r,k)}(t)$. Alternatively can process the data packets to promote them to the next state with rate $a_n^{(r,k)}(t)$. When $n \neq src(r)$, we do not consider source packet admission (i.e., $\lambda^{(r)}(t) = 0$).

The utility optimization problem **(P1)** is a convex optimization problem subject to linear constraints. For simplicity, we introduce a function $h(\cdot)$ of $y_n^{(r,k)}(t)$ to capture the flow conservation constraint in Eq. (1), given by,

$$h(y_n^{(r,k)}(t)) = \lambda^{(r)}(t)\mathbb{1}_{(k=1, n=src(r))} + \sum_{i \in \mathcal{N}} s_{i,n}^{(r,k)}(t)$$
$$+ a_n^{(r,k-1)}(t) - \sum_{j \in \mathcal{N}} s_{n,j}^{(r,k)}(t) - a_n^{(r,k)}(t). \tag{8}$$

Therefore, constraint (1) can be rewritten as

$$h(y_n^{(r,k)}(t)) \leq 0. \tag{9}$$

Using Eq. (9), we rewrite the queue update Eq. (2) as

$$Q_n^{(r,k)}(t+1) = Q_n^{(r,k)}(t) + h(y_n^{(r,k)}(t)). \tag{10}$$

Further, we simplify the sum of the utility function in Eq. (7) as $\sum_{r=1}^{R} U_r(\cdot)$ and rewrite utility maximization problem

$$\max_{\boldsymbol{h(y_n(t))}} \sum_{r=1}^{R} V \cdot U_r(t) \tag{11}$$

$$s.t. \quad h(y_n^{(r,k)}(t)) \leq 0, \quad \forall n \in \mathcal{N}, \tag{12}$$
$$(3), (4), (5), (6).$$

In this formulation, we introduced a positive term $V \geq 0$, which can be seen as a scaling parameter that controls the trade-off between the achievable utility and queue backlogs. The details are given in the performance analysis section. In order to solve this non-linear convex problem, we introduce the *Lagrange multiplier* $\theta_n^{(r,k)}(t) \in \mathbb{R}^{\mathcal{R} \times \mathcal{K}}$. The *Lagrange dual* problem is written as

$$\max \mathcal{L}_g(\cdot) = \sum_{r=1}^{R} V \cdot U_r(t) - \sum_{\substack{(r,k) \in (\mathcal{R}, \mathcal{K}),\\ n \in \mathcal{N}}} \theta_n^{(r,k)}(t) \cdot h(y_n^{(r,k)}(t))$$
$$s.t. \quad y_n \in \mathcal{A}, \tag{13}$$

where we simplify the constraints by letting $(3), (4), (5), (6) = \mathcal{A}$, (i.e., network capacity region). The *Lagrange dual* problem can be decoupled into sub-problems based on the decision variables.

$$\textbf{(SP1)} \max \sum_{\substack{(r,k) \in (\mathcal{R}, \mathcal{K}),\\ n = src(r)}} \left[V \cdot U_r(t) - \theta_n^{(r,k)}(t) \cdot h(y_n^{(r,k)}(t)) \right] \tag{14}$$

In **(SP1)**, we consider that the computing network admits packets at $src(r)$ node and with a packet state being $k = 1$, which is a special case. Therefore, we decouple the flow admission problem in **(SP1)**, where we consider $n = src(r), k = 1$, and the source rate given by $\lambda^{(r)}(t)$ at each slot. In other cases, where we do not consider the flow packet admission, the *Lagrange dual* in Eq. (13) includes the *internal forwarding* (i.e., $a_n^{(r,k)}(t)$) and *external forwarding* (i.e., $s_{n,j}^{(r,k)}(t)$) of packets.

[3]In our paper, we use non-negative utility function $\log(1+\lambda^{(r)}(t))$, leading to a proportionally fair allocation $\sum_{r=1}^{R} \frac{\overline{\lambda^*}^{(r)} - \lambda^{(r)}}{\overline{\lambda^*}^{(r)} + 1} \geq 0$. where $\overline{\lambda^*}^{(r)}$ is a proportionally fair operating point and $\lambda^{(r)}$ is any other feasible operating point. This also ensures that the utility function is well-defined with $\lambda^{(r)} \geq 0$ [39].

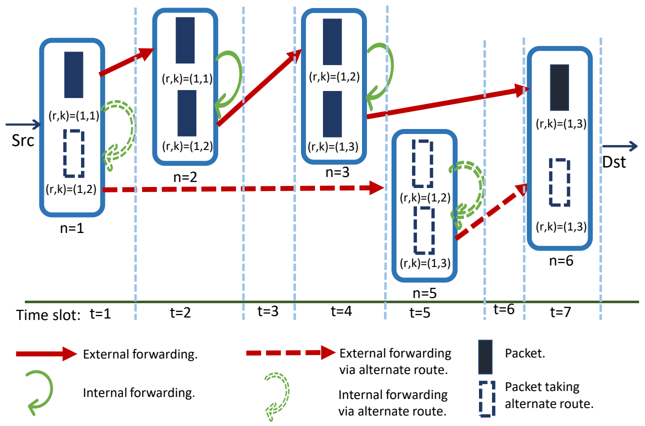


Fig. 3: We demonstrate internal/external forwarding of a packet along two alternate routes at each time slots. (a) A packet (e.g., (r,k): flow r and in state k) is externally forwarded to **node n=1** (solid red arrow) at slot $t = 1$, internally forwarded by **n=2** (solid green curved arrow) at slot $t = 2$. Then the packet is externally forwarded to **n=3** and further internally forwarded in **n=3** at slot $t = 4$. (b) Alternatively, the same packet (dashed) follows a different route. For example, the packet is internally forwarded in **node n=1** (dashed curved green arrow) at $t = 1$, externally forwarded to **n=5** (dashed red arrow) at $t = 5$, and delivered to the destination.

The *internal/external forwarding* couples a joint scheduling and routing problem which is given by,

$$\textbf{(SP2)} \quad \max \; - \sum_{\substack{(r,k)\in(\mathcal{R},\mathcal{K}),\\ n\in\mathcal{N}}} \theta_n^{(r,k)}(t)\cdot h(y_n^{(r,k)}(t))$$
$$= \max \sum_{\substack{(r,k),\\ n}} a_n^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\Big]$$
$$+ \sum_{\substack{(r,k),\\ n}} s_{n,j}^{(r,k)}(t) \sum_{j\in\mathcal{N}} \Big[\theta_n^{(r,k)}(t) - \theta_j^{(r,k)}(t)\Big]. \quad (15)$$

Proof. **Appendix A.** □

We simplify the summation $\sum_{\substack{(r,k)\in(\mathcal{R},\mathcal{K}),\\ n\in\mathcal{N}}}(\cdot) = \sum_{\substack{(r,k),\\ n}}(\cdot)$. We can solve the resource allocation problem in **(SP2)** if we can estimate the value of *Lagrange multiplier* (i.e., $\theta_n^{(r,k)}(t)$) from Eq. (15). Updating $\theta_n^{(r,k)}$ according to a queue dynamics (i.e., as $\theta_n^{(r,k)}(t) = Q_n^{(r,k)}(t)$) from Eq. (2), provides us an estimation. (The details can be found in **Appendix D.**) According to our analysis presented by Eq. (14), and Eq. (15), we provide the deterministic solutions to problem **(P1)** by decoupling it into sub-problems with readily available solutions. The sub-problems are solved sequentially and together they provide the joint algorithm to solve the initial problem **(P1)** as follows.

*1) **(SP1)**: **Flow Control**:* **(SP1)** isolates data packet admission problem in the source node $src(r)$ for each flow r.

$$\max_{\boldsymbol{\lambda^{(r)}(t)}} \sum_{\substack{(r,k=1),\\ n=src(r)}} \Big[V\cdot U_r(\lambda^{(r)}(t)) - Q_n^{(r,k)}(t)\cdot\lambda^{(r)}(t)\Big]$$
$$s.t. \quad \lambda^{(r)}(t) \in domain(U_r) \quad (16)$$

Remark 2. *Since AgI flows are independent,* $\max\sum(\cdot) = \sum\max(\cdot)$ *in **SP1**.*

The solution to **(SP1)** is given by,

$$\lambda^{(r^*)}(t) = \Big[U'^{-1}\big(\frac{Q_n^{(r,k)}(t)}{V}\big)\Big]_0^{\lambda_r^{max}} \quad (17)$$

The complexity of the step is $\mathcal{O}(|R|)$, where $|R|$ is the number of AgI flows in the network. We decouple the problem **(SP2)** into two sub-problems.

*2) **(SP2A)**: **Internal Forwarding**:* **(SP2A)** isolates the scheduling of computing resources at each computing node. Specifically, at slot t a particular (r,k) queue is scheduled to compute $a_n^{(r,k)}(t)$ amount of packets and promote the packets to queue $(r,k+1)$ at the same computing node, i.e.,

$$\max_{\boldsymbol{a_n^{(r,k)}(t)}} \sum_{\substack{(r,k)\in(\mathcal{R},\mathcal{K}),\\ n\in\mathcal{N}}} a_n^{(r,k)}(t)\Big[Q_n^{(r,k)}(t) - Q_n^{(r,k+1)}(t)\Big] \quad (18)$$
$$s.t. \quad (3),(4), \text{ and } a_n^{(r,k)}(t) \geq 0.$$

The solution to this sub-problem is given by the following,

$$a_n^{(r*,k*)}(t) = \arg\max_{\boldsymbol{(r,k)}} \Big[Q_n^{(r,k)}(t) - Q_n^{(r,k+1)}(t)\Big]\cdot\frac{\mathcal{C}_n}{\sigma^{(r,k)}\rho^{(r,k)}} \quad (19)$$

Note that, with $k = \mathcal{K}$, the computing function $(r,k+1)$ represents a dummy function and $Q_n^{(r,k+1)}$ is a dummy queue. This is a special case since the packets in $(r,k=\mathcal{K})$-th queue require no internal forwarding. The complexity of this step is $\mathcal{O}(|R\times K|)$, for eachnode $n\in\mathcal{N}$, where $|R|$ is the number of AgI flows in the network and $|K|$ is the number of required computing functions for each flow.

*3) **(SP2B)**: **External Forwarding**:* **(SP2B)** refers to the external packet forwarding between two neighboring computing nodes (i,j) by allocating communication resources. Here, a communication link is scheduled for routing the packets from data queue (r,k) residual to node n to a data queue (r,k) at downstream node j.

$$\max_{\boldsymbol{s_{n,j}^{(r,k)}(t)}} \sum_{(r,k)\in(\mathcal{R},\mathcal{K})} s_{n,j}^{(r,k)}(t)\sum_{j\in\mathcal{N}}\Big[Q_n^{(r,k)}(t) - Q_j^{(r,k)}(t)\Big] \quad (20)$$
$$s.t. \quad (5),(6) \text{ and } s_{n,j}^{(r,k)}(t) \geq 0.$$

we incorporate a maximum backpressure-based scheduling between the queues (in between the nodes to downstream) and available transmission rates at each time slot.

$$s_{n,j}^{(r^*,k^*)} = \arg\max_{\boldsymbol{\mu_{n,j}}} \sum_{n,j\in\mathcal{L}} \frac{\mu_{n,j}}{\sigma^{(r,k)}}\cdot w_{n,j}^{(r^*,k^*)}(t), \forall(n,j). \quad (21)$$

where,

$$w_{n,j}^{(r^*,k^*)}(t) \triangleq \arg\max_{\boldsymbol{(r,k)}}[Q_n^{(r,k)}(t) - Q_j^{(r,k)}(t)],$$
$$\forall(n,j),\forall(r,k), \quad (22)$$

where (n,j) denotes that computing node j (downstream) is in the communication range of computing node n and there exists an active communication link (n, j). The potential packet forwarding rate $\mu_{n,j}$ is given by the matrix $\boldsymbol{\mu} = [\boldsymbol{\mu_n}, ..]$ at each slot, where $\boldsymbol{\mu_n} = \{\mu_{n,j}\}$ represents a vector of potential packet forwarding rate at link $(n, j) \in \mathcal{L}$. It is important to note that, the actual rate is given by, $s_{n,j}^{(r^*,k^*)} = min[\frac{\mu_{n,j}}{\sigma^{(r,k)}}, Q_n^{(r,k)}(t) - Q_j^{(r,k)}(t)]$, and when $[Q_n^{(r,k^*)}(t) - Q_j^{(r,k^*)}(t)] \leq 0$, we set $\mu_{n,j} = 0$.

The complexity of this step is $\mathcal{O}(|R \times K||J|)$, where $|J|$ is the number of computing nodes that are in the neighborhood of node n. Different from the fixed-resource allocation-based counterparts, the proposed algorithm explores dynamic resource allocation via multi-path forwarding of AgI flow packets by augmenting the *internal/external forwarding*. An illustrative example is shown in Fig. 3 where computing nodes route a flow packet via alternative paths while externally/internally forwarding the packets.

V. Performance Analysis

To analyze the algorithm performance, we use the *Lyapunov-drift* equation which is defined as the quadratic of the queue backlog that measures the network congestion.

Definition 2. *(Lyapunov function) We define the Lyapunov function as a function of queue dynamics in the following*

$$L(t) = \frac{1}{2} \sum_{(r,k)} [Q_n^{(r,k)}(t)]^2, \tag{23}$$

and the Lyapunov-drift equation by,

$$(\Delta Q(t)) = L(t+1) - L(t). \tag{24}$$

If we can measure the upper bound of the drift $(\Delta Q(t))$, then we can establish an upper bound of the queue backlog. Using the definition of *Lyapunov-drift* equation (Eq. 24) and Eq. (10), we derive the following lemma that gives us the upper-bound of drift.

Lemma 1. *At each time slot t, the Lyapunov-drift equation can be finitely upper-bounded by,*

$$(\Delta Q(t)) \leq G - \epsilon \cdot Q_n^{(r,k)}(t)$$

We define two positive constant terms G and ϵ as

$$G \triangleq [\sum_{j \in \mathcal{N}} s_{n,j}^{max} + a_n^{max}]^2 + [\lambda_r^{max} + \sum_{i \in \mathcal{N}} s_{i,n}^{max} + a_n^{max}]^2,$$
$$\epsilon \triangleq [\sum_{j \in \mathcal{N}} s_{n,j}^{(r,k)} + a_n^{(r,k)}] - [\lambda^{(r)} + \sum_{i \in \mathcal{N}} s_{i,n}^{(r,k)} + a_n^{(r,k-1)}]. \tag{25}$$

Proof. Appendix E. □

Using the *Lyapunov drift* bound given by Lemma 1, we will prove the data queue stability and near-optimal utility of the proposed algorithm. We state the following assumptions that are critical to the analysis.

Assumption 2 (Infinite reservoir and real-value assumptions)**.**

1) *The transport layer queues are infinitely backlogged (infinite application demand). Therefore, there are always flow packets to admit in the network leading to $\lambda^{(r)}(t)$ being a non-zero quantity.*
2) *Moreover, we assume that $\lambda^{(r)}(t)$ can be a real value (not necessarily integer/integer multipliers of a given packet length). Let $\lambda^{(r)} \in [0, \lambda^{max}]$ and $\lambda^{(r)}(t)$ be an infinite sequence of non-negative real numbered values in the set. Thus, $\frac{1}{T}\sum_{t=0}^{T-1} \lambda^{(r)}(t) \in [0, \lambda^{max}]$. Specifically, we assume that there exists a continuous byte stream at the transport layer and $src(r)$ scales the size of the admitted data as a real-numbered value, e.g., 1.2 MBps. To summarize, the source rate is assumed to be a continuous real-numbered value at each time slot rather than a discrete integer value.*

Using the Lemma 1 and Assumption 2, we establish that our proposed algorithm maintains a finite time-average queue backlog and near-optimal network utility. First, we set up the necessary variables before stating the theorem.

- **Conditional Drift:** Let us represent a queue backlog process by $\overline{\boldsymbol{Q}}(t) = [Q_n^{(r,k)}(t)]$ that evolves according to some probability law and represents the current queue backlogs. Here, we slightly abuse the notation of queue backlog at a slot t to represent the queue backlog process. Recall, we defined the *Lyapunov function* by $L(Q_n^{(r,k)}(t)) = [Q_n^{(r,k)}(t)]^2$ as a quadratic function. For the given queue backlog vector $\overline{\boldsymbol{Q}}(t)$, we define the *conditional Lyapunov drift* $\Delta(\overline{\boldsymbol{Q}}(t))$ as

$$\Delta(\overline{\boldsymbol{Q}}(t)) \triangleq \mathbb{E}\{L(\overline{\boldsymbol{Q}}(t+1)) - L(\overline{\boldsymbol{Q}}(t)) \mid \overline{\boldsymbol{Q}}(t)\}. \tag{26}$$

- **Target Utility:** We denoted the network utility at each time slot by $U_r(\lambda^{(r)}(t))$ and assume, the achievable utility is upper-bounded by a finite value as $\sum_{r=1}^{R} U_r(\lambda^{(r)}(t)) \leq U^{max}$ for all t, where the utility function is a non-negative concave function. We assume that there exists a stationary randomized policy which gives the decision vector $\hat{y}_n[t] \triangleq \{\hat{y}_n^{(r,k)}(t)\}$ that deterministically satisfies $\hat{\lambda}_n[t] = \lambda^*$ and produces a utility value $U(\lambda^*)$.

Theorem 1. *Suppose there exist positive constants V acting as a utility control parameter and the constants ϵ, G as it is defined in Eq. (25) for all time slots t, and queue backlog process $\overline{\boldsymbol{Q}}(t)$. Further, by using the result of Lemma 1, if Lyapunov-drift satisfies the following drift-minus-utility bound,*

$$\Delta(\overline{\boldsymbol{Q}}(t)) - V \sum_{r=1}^{R} \mathbb{E}\{U_r(\lambda^{(r)}(t)) \mid \overline{\boldsymbol{Q}}(t)\}$$
$$\leq G - \epsilon \sum_{\substack{(r,k),\\ n}} Q_n^{(r,k)}(t) - VU(\lambda^*), \tag{27}$$

then the queue backlogs are upper-bounded by

$$\limsup_{T \to \infty} \sum_{t=0}^{T-1} \sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} \leq \frac{G + VU^{max}}{\epsilon}. \tag{28}$$

Further, assuming the data queues are initially empty, using both Assumption 2 and Assumption 1, the achievable utility of the proposed algorithm satisfies

$$\liminf_{T\to\infty} \sum_{r=1}^{R} U_r(\overline{\lambda}^{(r)}(T)) \geq \nu^* - \frac{G}{V}, \tag{29}$$

where we denote the optimal utility by ν^* which will be defined later in the proof.

Proof. The proof is given in Appendix F. □

VI. Experimental Evaluation

We evaluate the performance of the proposed algorithm by performing extensive simulations. First, we briefly discuss the simulation settings and then we present our experimental results. We perform our experiments considering a wireless computing network with $|\mathcal{N}| = 6$ computing nodes. The network is connected via directed wireless links between the computing nodes. In particular, we focused our experiments on a cell-partitioned network topology where we assume each cell contains a single node. The communication occurs among the nodes residing in adjacent cells that use orthogonal frequency bands. We also assume that the cell structure is rectilinear. It is important to mention that, the cell structure is not critical to our analysis. However, the assumption simplifies the exposition and decouples the routing decisions cell-by-cell. We assume the communication link capacities vary between $10 - 19 packets/slot$. We use $log(1 + \lambda^{(r)})$ as our system utility metric. (See Table II). The experimental results are given below.

TABLE II: Simulation Setup

Parameter	Numerical argument
Number of Computing Nodes	$\|\mathcal{N}\| = 6$
Number of flows	$\|\mathcal{R}\| = 3$
Number of compute functions	$\|\mathcal{K}\| = 3$
Maximum source rates	$\lambda_r^{max} = [9, 8, 7] MBps$
CPU Scaling Factor	$f = [1, 1.3, 1.7, 2]$
Utility scaling parameter	$V = [0, 250, 500, ..., 30000, 45000, 50000]$
Computing capacity	$\mathcal{C}_{(n=1\to 6)} = [145, 175, 160, 130, 175, 140]$ (CPU cycles/slot)
Communication Capacity	$\mu = [10 - 19] MBps$
Flow 1	
Required CPU cycles (with f=2)	$\rho^{(r=1,k=1\to 3)} = [3.5, 7, 14]$ (Cycles/MB)
Packet size	$\sigma^{(r=1,k)} = 3$ (MB)
Flow 2	
Required CPU cycles (with f=2)	$\rho^{(r=2,k=1\to 3)} = [3, 6, 12]$ (Cycles/MB)
Packet size	$\sigma^{(r=2,k)} = 3.5$ (MB)
Flow 3	
Required CPU cycles (with f=2)	$\rho^{(r=3,k=1\to 3)} = [5, 10, 20]$ (Cycles/MB)
Packet size	$\sigma^{(r=3,k)} = 3$ (MB)

A. Comparison with state-of-art

We compare with the existing works to show the persuasiveness of our proposed algorithm. The baselines are introduced in the following.

1) **Optimal Policy:** The optimal policy assumes the optimal utility.
2) **Proposed algorithm:** Our proposed algorithm investigates the internal/external forwarding of AgI flow packets via augmenting the backpressure scheduling and routing.

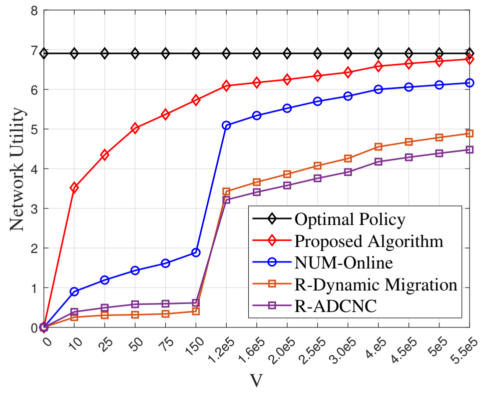
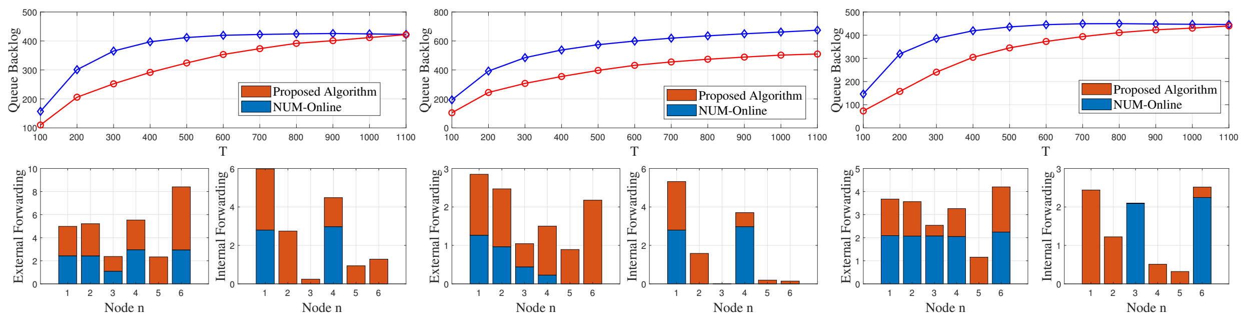


Fig. 4: Network utility under different V parameters.

3) **NUM-Online [10]:** Proposes a fairness-based flow control and resource allocation an NFV-based scenario. The work assumes a fixed computing/communication resource allocation configuration to flows in the computing nodes and devises a flow control mechanism.
4) **R-Dynamic Migration [12]:** This method is a reduced variant of the algorithm proposed by [12] that assumes a known source rate, involves computing/communication resource allocation, and computing function migration by halting the processing. We measure their utility by injecting a time-varying source rate.
5) **R-ADCNC [13]:** Considers a cost/delay associated with the computing function reconfiguration process. To compare, we consider the delay by halting the source admission during function reconfiguration. We measure their utility by injecting a time-varying source rate.

1) Network utility: We compare the network utility achieved by our proposed algorithm with respect to the baseline algorithms in Fig. 4. Compared to the existing works, *internal/external* forwarding does not affect the flow admission and scheduling/routing of any other flow packets. The proposed algorithm provides a joint algorithm for flow control, scheduling, and routing by streamlining *cross-layer control* and convex duality and guarantees superior performance in terms of network utility. Our experiment cross-verifies the analysis in Theorem 1 and achieves close-to-optimal network utility with a vanishing utility gap of $\frac{G}{V}$. When the value of V increases from 150 to $1.2e5$, the network utility of the NUM-Online, R-Dynamic Migration, and R-ADCNC algorithms shows significant improvements. However, the proposed algorithm increases steadily for small V values, e.g., $V = [10, 25, 50, 150]$ and quickly converges. It is also important to note that, unlike the baselines our proposed algorithm maintains a trade-off between data queue stability and utility optimality.

2) Queue backlog and internal/external forwarding tradeoff: Since the NUM-Online [10] is the second best performer, we compare NUM-online in terms of data queue congestion and joint resource allocation. In Fig. 5, we show the comparison between NUM-Online and our Proposed algorithm in terms of data queue backlogs and *internal/external*



(a) *Flow 1* (b) *Flow 2* (c) *Flow 3*

Fig. 5: Queue backlogs and internal/external forwarding at each computing node.

forwarding rates at each computing node. To perform this experiment, we fix $V = 1000$ and iterate the algorithm over $T = 1100$ time slots. NUM-Online [10] proposes a fixed deployment of computing/communication resources at particular computing nodes. On the other hand, our proposed algorithm augments *internal/external forwarding* to allocate computing/communication resources and steer the flow packets via different routes toward the destination. Therefore, our proposed algorithm experiences lower data queue backlogs for each flow (see the top graph).

Additionally, our algorithm derives internal forwarding rate $a_n^{(r,k)}(t)$ and external forwarding rate $s_n^{(r,k)}(t)$ by solving the **SP2**. Therefore, the algorithm maintains a proper trade-off between the computing/communication resources in the network. The bar chart (red color) in Fig. 5a shows that the average internal/external forwarding rates of *flow 1* packets given by the proposed algorithm where the *flow 1* packets are collaboratively computed and forwarded by all computing nodes.

It is important to mention that NUM-Online [10] does not provide an *internal/external packet forwarding* strategy. Rather, NUM-Online offers predetermined computing/processing of a flow at specific computing nodes and fixed path routing. We compare the computing/communication resource allocation rates by NUM-Online at each computing node(shown in blue bars) with our proposed backpressure algorithm. Both our analysis and experimental evaluation show that NUM-Online provides an uneven resource allocation (see blue color) to the packets at different computing nodes.

B. *Experiment with varying parameters*

We test our proposed algorithm by varying different parameters, e.g., utility scaling V and maximum source rate λ^{max} requirements of each flow, packet size $\sigma^{r,k}$, and CPU scaling factor denoted as $\rho^{(r,k)}$. The results are given below.

1) Varying maximum source rate: We show that our proposed algorithm maintains resource fairness among different AgI flows in Fig. 6. Specifically, we performed the experiment with different $\boldsymbol{\lambda}^{max} = [12, 6, 3]$ (i.e., *flow1* with $\lambda^{max} = 12$) for each flow instead of a fixed λ^{max}. We show the achievable utilities of *flow1-flow3* in the network under different values of the scaling parameters V. The utility of each individual

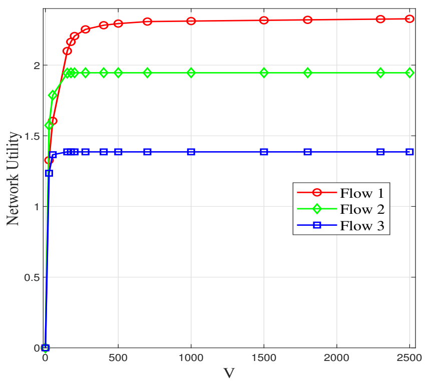
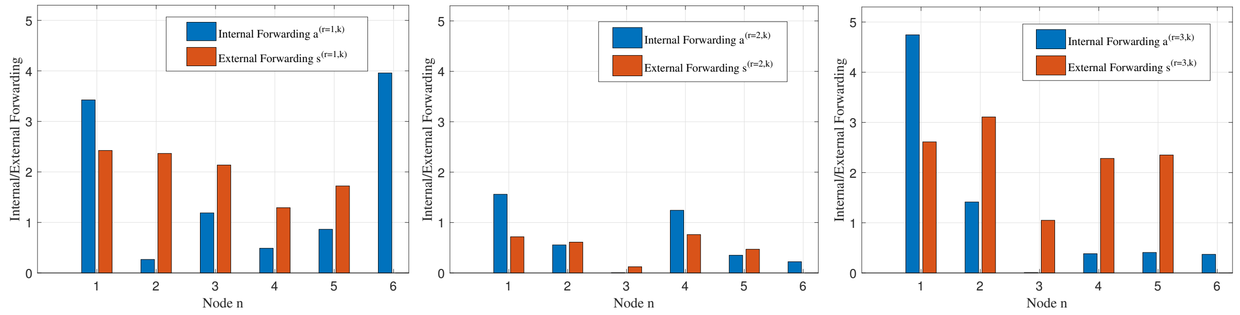


Fig. 6: Network utility of different flows.

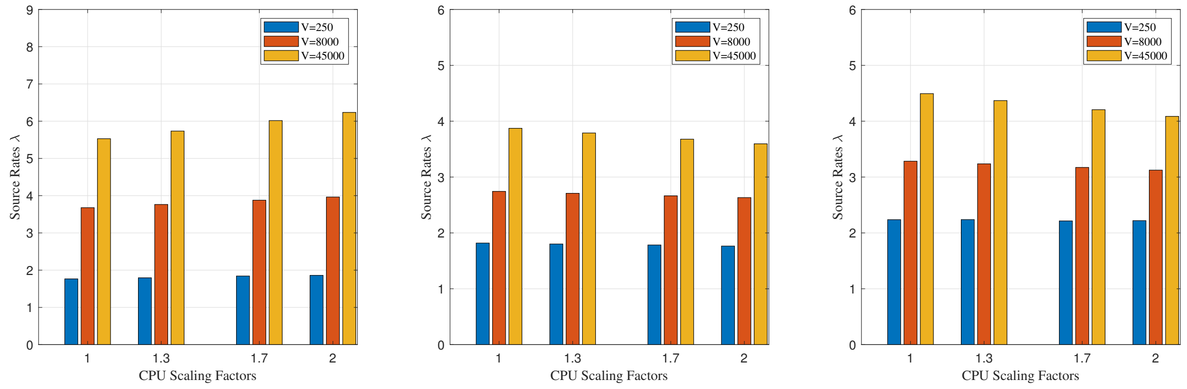
flow sharply increases with the increment of V value and with larger values, the utilities converge as the source rates converge while maintaining fairness among different network flows.

2) Varying packet size: We show the effect of change in packet sizes in Figs. 7a-7c. For this particular experiment, we consider that the packet length changes after being processed by a function (r, k). For example, the packet size may increase when a function adds extra payload information to the packet. In this experiment, we set the revised packet lengths of *flow 1* as $\sigma^{(r=1,k=1\to3)} = [5.5, 4, 6.5]MB$ after being processed by k-th computing function. Similarly, we set $\sigma^{(r=2,k=1\to3)} = [5.5, 6.5, 6]MB$ and $\sigma^{(r=2,k=1\to3)} = [2, 2, 1.5]MB$. The change in packet lengths at different packet states implies that the packet requires different computing/communication resources at each state. This change affects the overall internal/external forwarding rates of a particular flow packet at different computing nodes. For example, due to the larger packet length and higher computing demand of *flow 2* packets, computing nodes can internally/externally forward the packets of *flow 2* at a lower rate compared to *flow 3* and *flow 1*. Especially, *flow 3* packets have shorter packet lengths, and therefore packets are processed by computing node $n = 1$ at a higher rate than *flow 1* and *flow 2*. *Flow 1* packets are mostly externally forwarded and then processed by node $n = 6$ which shows that our proposed algorithm maintains a trade-off by distributing compute/communication resources allocations



(a) *Flow 1* (b) *Flow 2* (c) *Flow 3*

Fig. 7: Internal/external forwarding at computing nodes under packet size changing case.



(a) *Source rate of flow 1.* (b) *Source rate of flow 2.* (c) *Source rate of flow 3.*

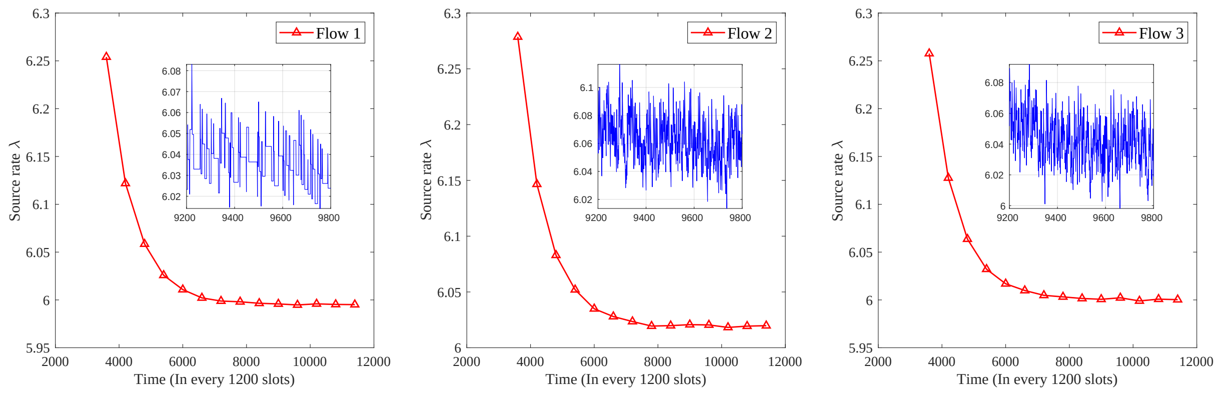
Fig. 8: Comparison of source rates under different CPU scaling factors and V.

across the computing network. Note that, we do not consider any flow scaling (e.g., change in the number of packets after computation) therefore the total number of packets remains the same before being delivered.

3) Varying CPU scaling factor: We show the effect of computation resource requirements (required CPU cycles) variation to process the packets (i.e., $\rho^{(r,k)}$), on achievable source rate $\lambda^{(r)}$. We incorporate a parameter of the CPU scaling factor (denoted by f) [10], that is the variation of required CPU cycles in the sequence of functions of an AgI flow r. In practice, the resource requirements are different from specific functions. Assume, the flow r requires processing by the system firewall and then by an encryptor which requires more CPU cycles than a network firewall. To simplify our study on the effects of computing resource requirements on $\lambda^{(r)}$, we assume the subsequent functions require more computation resources than their previous functions, e.g., $\rho^{(r,k)}<\rho^{(r,k+1)}$. It is important to note that, this setting is to mimic the workload variation of the functions of an AgI flow. In practice, we replace this particular setting with a random process. To illustrate the variations, we use a finite scaling factor, e.g., $\rho^{(r,k)} * f = \rho^{(r,k+1)}$, meaning that processing of $(r, k+1)$ requires f times the CPU cycles than (r, k).

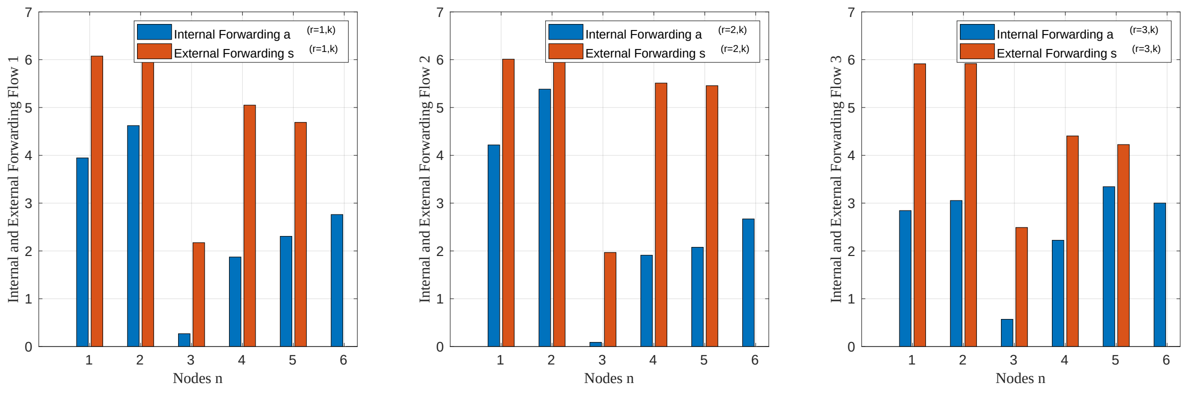
In Fig. 8, we present the effects of the scaling factor for each flow. Specifically, we set the computing resource requirement of $k = 1$-st function of the *flow 1* as $\rho^{(r=1,k=1)} = 3.5$ with a set of different scaling factors f. For example, with scaling factor $f = 2.0$ and $\rho^{(r=1,k=1)} = 3.5$, the required CPU cycles to process each bit of a *flow 1* packet is given by a vector $\boldsymbol{\rho}^{(r=1,k)} = [3.5, 7, 14]$ for each functions $k \in \mathcal{K}$. We set the initial $\boldsymbol{\rho}^{(r=3,k)} = [5, 8, 7]$ for this particular experiment. It is evident from the experimental result that incremental CPU scaling factors tend to lower the source rate of *flow3*. To focus on the effect of the CPU-scaling factor on source rates, we assume that the processing of a packet in (r, k) does not change the packet size in this particular experiment. Rather, our goal is to show only the effects on the source rates. We set packet lengths vector as $\boldsymbol{\sigma}^{(r)} = [3, 3.5, 3]$ for the three flows.

Among all the admissible flows into the network, *flow 1* and *flow 2* require lower computing resources to process a packet, and the packet sizes are similar. Therefore, they offer similar source rates. In particular, with large V, such as $V = 45000$, the source rates of both of these flows are the same. However, with a smaller value of this parameter $V = 250$, *flow 2* achieves a better source rate. This is due to the different internal/external forwarding rates of flow r in the computing nodes. When required computing resources to perform function-specific processing of a packet increase by a factor, (e.g., $f = 1.5$), the packets have to wait longer in a queue, and the backlog increases. When the backlog at the



(a) *Source rate of Flow1.* (b) *Source rate of Flow2.* (c) *Source rate of Flow3.*

Fig. 9: Convergence of flow packet admissions.



(a) *Internal-external forwarding rates of Flow 1.* (b) *Internal-external forwarding rates of Flow 2.* (c) *Internal-external forwarding rates of Flow 3.*

Fig. 10: The comparison between internal forwarding and external forwarding rates in each computing node of the network.

computing nodes is large, the proposed algorithm lowers the source rates of the specific flow at $src(r)$. On the other hand, the parameter V also regulates the source rate $\lambda^{(r)}$. Overall, there is a trade-off between the queue backlog, achievable source rates, and V. When V becomes larger, the source rates $\lambda^{(r)}$ increase. With an increment of the computing resource requirements to process each packet, the source rates decrease. The effects of these parameters are illustrated in Fig. 8.

C. Long-term performance

We test the performance of our proposed algorithm in the long term. Specifically, we take the running average of source rates of each flow and internal/external forwarding rates at each node in a predefined time window and show the convergence of source rates and the overall internal/external forwarding trade-off over time.

1) Source rate: Our proposed algorithm stabilizes the network utility to a level while maintaining fairness among the different flows. We experiment on the convergence of network utilities over $T = 12000$ time slots with a fixed $V = 20000$. For better understanding and presentation, we calculate the time-average flow rates in small time-windows (during every $t = 600$ time slots), to show the change in achievable utility over time. Since at the beginning of the iteration, the source queues of the networks are empty (i.e., $Q_{n=1}^{(r,k=1)}(0) = 0$), therefore the queues can accept more packets from the *transport-layer reservoirs*. The parameter V regulates how many packets a particular source queue can accept. When the V is very large, the queue (i.e., $Q_{n=1}^{(r,k=1)}(t)$) can allow more source packets and add them to the backlogs. Therefore, with sufficiently large enough V, a source queue can accept the number of packets to its maximum limit, i.e., λ_r^{max}. With time, the queue backlogs become large since the computing nodes can accommodate a limited number of internal/external packet forwarding. Therefore, to avoid congestion and instability, the source nodes $src(r)$ regulate the flow of the packets over time. As a result, the source rate in the network decreases over time until the rate converges to a stable point. The simulations presented in Fig. 9a-9c, demonstrate the aforementioned trends in source rates. Along with the time-average rate, we present a detailed speculation of the source rate dynamics over each time slot (e.g., $[9200 \rightarrow 9800]$). The dynamics in the region show that source rates do not fluctuate significantly, varying at most a range of 0.10.

2) Trade-off in internal/external forwarding rates: Specifically, the internal/external forwarding rates evolve depending on a metric that is based on the computing demands, computing resources in a computing node, and queue backlogs at each time slot. Since provided communication resources to communication links are different (e.g., due to the power assignment on a link), the achievable external forwarding rates are different across the computing network. Similarly, the processing requirements of packets are different for each (r,k) tuple in the vector $\boldsymbol{\rho}^{(r,k)}$. Considering all the facts, the proposed algorithm determines the external/internal forwarding rates at each time slot. Moreover, there is an interplay between the internal forwarding and external forwarding rates. Computing nodes perform both internal and external forwarding according to available resources and backpressure differences. We perform a comparative experiment on internal/external packet forwarding rates (shown in Fig. 10) at each computing node to illustrate this interplay. In our experiment, we set node $n = 3$ with the highest communication resources (e.g., the outgoing communication links from $n = 3$ accommodate high data rate $\mu_{n,j}$) among all the computing nodes. Therefore, to best use its resources, node $n = 3$ chooses to perform external packet forwarding rather than internal forwarding.

We performed the simulation with $\boldsymbol{\rho}^{(r,k)} = [2,3,4,3,3,3,5,8,7](inCPUcycle/bit)$, $V = 450000$, and $\boldsymbol{\mathcal{C}}_{(n=1\to 6)} = [145,175,160,130,175,140]$ in CPU cycle). The computing nodes perform the external forwarding according to the solution in Eq. (21). The potential link rates $\mu_{n,j}$ are given by the matrix $\boldsymbol{\mu}$ at a time slot. In this particular case, we fix some arbitrary rate for $\mu_{n,j}$ varying from $12 \to 19$ $packets/slot$, based on whether a directed wireless link exits between nodes (n,j), and 0 otherwise, and $\mu_{n,n} = 0$. We inspect that node $n = 3$ performs more external forwarding of packets than compared to internal forwarding due to its higher communication resources. Each of the flows takes the advantages of computing and communication resources from the nodes $n \in \mathcal{N}$ differently according to the proposed algorithm, which proves the correctness and applicability of the proposed algorithm in a wireless computing network.

VII. Concluding Remarks

We studied a dynamic scheduling and routing problem in the wireless computing network via packet forwarding strategies under an NUM framework. The strategies involve efficient resource allocation decisions with minimum information exchange that develops an augmented backpressure-based algorithm. In the future, we will consider studying the storage resource allocation with scheduling and routing in AgI flows under mobility constraints.

References

[1] Q. Duan, S. Wang, and N. Ansari, "Convergence of networking and cloud/edge computing: Status, challenges, and opportunities," *IEEE Network*, vol. 34, no. 6, pp. 148–155, 2020.

[2] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.

[3] Y. Chen, J. Wu, and B. Ji, "Optimizing flow bandwidth consumption with traffic-diminishing middlebox placement," in *Proc. of 49th International Conference on Parallel Processing-ICPP*, virtual event, 2020.

[4] M. Shi, X. Lin, and L. Jiao, "Power-of-2-arms for bandit learning with switching costs," in *Proc. of ACM MobiHoc*, Seoul, South Korea, 2022.

[5] X. Zhou, Y. Gao, C. Li, and Z. Huang, "A multiple gradient descent design for multi-task learning on edge computing: Multi-objective machine learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 121–133, 2022.

[6] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal control of wireless computing networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 8283–8298, 2018.

[7] M. Barcelo, J. Llorca, A. M. Tulino, and N. Raman, "The cloud service distribution problem in distributed cloud networks," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 344–350.

[8] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *Proc. of IEEE INFOCOM*, Atlanta, GA, USA, 2017.

[9] J. Li, W. Shi, Q. Ye, S. Zhang, W. Zhuang, and X. Shen, "Joint virtual network topology design and embedding for cybertwin-enabled 6G core networks," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 313–16 325, 2021.

[10] L. Gu, D. Zeng, S. Tao, S. Guo, H. Jin, A. Y. Zomaya, and W. Zhuang, "Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1059–1071, 2019.

[11] J. Zhang, A. Sinha, J. Llorca, A. M. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1760–1773, 2021.

[12] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic flow migration for embedded services in SDN/NFV-enabled 5G core networks," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2394–2408, 2020.

[13] C.-H. Wang, J. Llorca, A. M. Tulino, and T. Javidi, "Dynamic cloud network control under reconfiguration delay and cost," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 491–504, 2019.

[14] Q. Liu, H. Zeng, and M. Chen, "Network utility maximization under maximum delay constraints and throughput requirements," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2132–2145, 2020.

[15] X. Lin and N. B. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 766–781, 2006.

[16] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected areas in Communications*, vol. 24, no. 8, pp. 1452–1463, 2006.

[17] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, 2006.

[18] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[19] M. J. Neely and R. Urgaonkar, "Optimal backpressure routing for wireless networks with multi-receiver diversity," *Ad Hoc Networks*, vol. 7, no. 5, pp. 862–881, 2009.

[20] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-aware dynamic SFC mapping and scheduling in SDN/NFV-enabled space–air–ground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5824–5838, 2022.

[21] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded vnf chains in 5g core networks," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 692–704, 2018.

[22] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *2015 11th International Conference on Network and Service Management (CNSM)*, 2015, pp. 50–56.

[23] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2118–2131, 2018.

[24] ——, "Dynamic network service optimization in distributed cloud networks," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016, pp. 300–305.

[25] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation scheduling, caching, and communication in data-intensive computing networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1058–1072, 2021.
[26] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Joint compute-caching-communication control for online data-intensive service delivery," *IEEE Transactions on Mobile Computing*, 2023.
[27] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 2, 2004, pp. 1484–1489 Vol.2.
[28] I.-H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Transactions on Wireless Communications*, vol. 14, no. 2, pp. 1071–1079, 2014.
[29] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.
[30] I.-H. Hou and P. Kumar, "Utility maximization for delay constrained QoS in wireless," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.
[31] A. Sinha and E. Modiano, "Optimal control for generalized network-flow problems," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 506–519, 2017.
[32] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Decentralized control of distributed cloud networks with generalized network flows," *IEEE Transactions on Communications*, vol. 71, no. 1, pp. 256–268, 2022.
[33] ——, "Ultra-reliable distributed cloud network control with end-to-end latency constraints," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2505–2520, 2022.
[34] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1047–1060, 2020.
[35] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Mobile edge computing network control: Tradeoff between delay and cost," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.
[36] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing throughput of delay-sensitive NFV-enabled request admissions via virtualized network function placement," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1535–1548, 2019.
[37] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, "QoS-aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–27, 2020.
[38] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, and J. Chen, "Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4247–4262, 2021.
[39] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions On Networking*, vol. 16, no. 2, pp. 396–409, 2008.
[40] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, pp. 237–252, 1998.
[41] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.
[42] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010. [Online]. Available: https://ieeexplore.ieee.org/document/6813406
[43] S. Sarkar and L. Tassiulas, "End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1246–1259, 2005.

Appendix A
Proof of Eq. (15)

Proof. Since Sub-problem **SP2** considers the *internal/external forwarding* of packets, we omit the term related to the source rate $\lambda^{(r)}$ in the proof. Eq. (15) states the following.

$$\max \quad - \sum_{\substack{(r,k)\in(\mathcal{R},\mathcal{K}),\\ n\in\mathcal{N}}} \theta_n^{(r,k)}(t)\cdot h(y_n^{(r,k)})(t)$$

Using the definition of $h(y_n^{(r,k)})(t)$ from Eq. (8) and get

$$\sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\Bigg[\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) + a_n^{(r,k)}(t) - \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t) - a_n^{(r,k-1)}(t)\Bigg].$$

Further, we distribute $\theta_n^{(r,k)}(t)$ and decouple the terms related to *internal* and *external* forwarding.

$$\begin{aligned}
&\sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\Big[a_n^{(r,k)}(t) - a_n^{(r,k-1)}(t)\Big] \\
&\quad + \sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\Bigg[\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\Bigg] \\
&\overset{(31)}{=} \sum_{\substack{(r,k),\\ n}} a_n^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\Big] \\
&\quad + \sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\Bigg[\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\Bigg]. \qquad (30)
\end{aligned}$$

Here, the first part of Eq. (30) follows due to Eq. (31) given by the result of **Lemma 2**. Further, we distribute $\theta_n^{(r,k)}(t)$ in the second term to yield,

$$\begin{aligned}
&\sum_{\substack{(r,k),\\ n}} a_n^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\Big] \\
&+ \Bigg[\sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - \sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\Bigg] \\
&\overset{(33)}{=} \sum_{\substack{(r,k),\\ n}} a_n^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\Big] \\
&\quad + \sum_{\substack{(r,k),\\ n}} \sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_j^{(r,k)}(t)\Big].
\end{aligned}$$

Here, where the final result follows due to Eq. (33) given by **Lemma 3** and concludes the proof. ☐

Appendix B
Proof of Eq. (31)

Lemma 2. ***(Internal Forwarding)***

$$\begin{aligned}
&\sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\Big[a_n^{(r,k)}(t) - a_n^{(r,k-1)}(t)\Big] \\
&= \sum_{\substack{(r,k),\\ n}} a_n^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\Big], \forall n\in\mathcal{N}, \forall r\in\mathcal{R}.
\end{aligned} \qquad (31)$$

Proof. Fix a node n, and a flow r. Assume there exists a dummy computing function, $(r, k = 0)$, at the beginning of the sequence (r, k) that does not compute packets and internally forward the packets at rate $a_n^{(r,k=0)}(t)$ to queue $(r, k = 1)$. The product of internal forwarding rate and the *Lagrange multiplier* associated with this function is $\theta_n^{(r,k=0)}(t)\, a_n^{(r,k=0)}(t)$.

Similarly consider another dummy function $(r, |\mathcal{K}|+1)$ that is receiving packets at rate $a_n^{(r,|\mathcal{K}|)}(t)$ from queue $(r, |\mathcal{K}|)$. The product of associated *Lagrange multiplier* and rate of this dummy function is given by $\theta_n^{(r,|\mathcal{K}|+1)}(t)\ a_n^{(r,|\mathcal{K}|)}(t)$. Adding $\theta_n^{(r,k=0)}(t)\ a_n^{(r,k=0)}(t)$ and subtracting $\theta_n^{(r,|\mathcal{K}|+1)}(t)\ a_n^{(r,|\mathcal{K}|)}(t)$ in the left-hand of Eq. (31) yields,

$$\sum_{k\in\{0,..\mathcal{K}\}} \theta_n^{(r,k)}(t)a_n^{(r,k)}(t) - \sum_{k\in\{1,..\mathcal{K}+1\}} \theta_n^{(r,k)}(t)a_n^{(r,k-1)}(t)$$

Expanding the summations we have the following.

$$\begin{aligned}&\big[\theta_n^{(r,k=0)}(t)a_n^{(r,k=0)}(t) + \theta_n^{(r,k=1)}(t)a_n^{(r,k=1)}(t)\\ &+ ... + \theta_n^{(r,k=|\mathcal{K}|)}(t)a_n^{(r,k=|\mathcal{K}|)}(t)\big]\\ &- \big[\theta_n^{(r,k=1)}(t)a_n^{(r,k=0)}(t) + \theta_n^{(r,k=2)}(t)a_n^{(r,k=1)}(t)\\ &+ ... + \theta_n^{(r,k=|\mathcal{K}|+1)}(t)a_n^{(r,k=|\mathcal{K}|)}(t)\big]\end{aligned}$$

Since $k \in \{0, ..\mathcal{K}\}$ and $k \in \{1, ..\mathcal{K}+1\}$ have equal cardinality, we take element-wise difference,

$$\begin{aligned}&\big[\theta_n^{(r,k=0)}(t)a_n^{(r,k=0)}(t) - \theta_n^{(r,k=1)}(t)a_n^{(r,k=0)}(t)\big]\\ &+ \big[\theta_n^{(r,k=1)}(t)a_n^{(r,k=1)}(t) - \theta_n^{(r,k=2)}(t)a_n^{(r,k=1)}(t)\big]\\ &\quad ..... \quad ..... \quad .....\\ &+ \big[\theta_n^{(r,k=|\mathcal{K}|)}(t)a_n^{(r,k=|\mathcal{K}|)}(t) - \theta_n^{(r,k=|\mathcal{K}|+1)}(t)a_n^{(r,k=|\mathcal{K}|)}(t)\big].\end{aligned}$$

Summarizing this, we have

$$\sum_k a_n^{(r,k)}(t)\big[\theta_n^{(r,k)}(t) - \theta_n^{(r,k+1)}(t)\big]. \tag{32}$$

Since Eq. (32) holds for all r at node n, it completes the proof. □

APPENDIX C
PROOF OF EQ. (33)

Lemma 3. ***(External Forwarding)***

$$\begin{aligned}&\Bigg[\sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - \sum_{\substack{(r,k),\\ n}} \theta_n^{(r,k)}(t)\sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\Bigg]\\ &= \sum_{\substack{(r,k),\\ n}}\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t)\theta_n^{(r,k)}(t) - \sum_{\substack{(r,k),\\ n}}\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t)\theta_j^{(r,k)}(t)\\ &= \sum_{\substack{(r,k),\\ n}}\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t)\Big[\theta_n^{(r,k)}(t) - \theta_j^{(r,k)}(t)\Big],\\ &\qquad \forall n\in\mathcal{N}, \forall (r,k)\in(\mathcal{R},\mathcal{K}).\end{aligned} \tag{33}$$

Proof. The proof is followed by algebraic manipulations as shown in [41]. □

APPENDIX D
RELATION BETWEEN QUEUE LENGTH AND *Lagrange multiplier*

Recall, **SP1** describes the flow control problem, i.e., $\lambda^{(r)}(t)$ and **SP2** describes the ***joint internal/external forwarding***. If the associated ***Lagrange multiplier*** vector is known, solutions to the sub-problems are readily available. Therefore, the algorithm is fully specified if we can estimate the ***Lagrange multiplier*** vector. A natural way of estimating the ***Lagrange multiplier*** is similar to the dual algorithm for the internet given in the following,

$$\begin{aligned}\theta_n^{(r,k)}(t+1) = \Bigg[\theta_n^{(r,k)}(t) + \epsilon\cdot\Bigg[\lambda^{(r)}(t) + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\\ + a_n^{(r,k-1)}(t) - \sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - a_n^{(r,k)}(t)\Bigg]\Bigg]^+,\end{aligned} \tag{34}$$

where $\lambda^{(r)}(t)$ is the solution of **SP1** and solution to **SP2** is given by $a_n^{(r,k)}(t)$ and $s_{n,j}^{(r,k)}(t)$ from Eq. (15). Note that, Eq. (34) is the *wireless counterpart* of the differential equation in the dual algorithm for the Internet with a constant step size ϵ. In our algorithm, we assume $\epsilon = 1$. The behavior of $\theta_n^{(r,k)}(t)$ is similar to the queue length update for each data queue indexed by (r, k) at node n. Therefore, we estimate the *Lagrange multiplier* by considering $\theta_n^{(r,k)}(t) = Q_n^{(r,k)}(t)$. By our analysis of how we collect the difference between the ***Lagrange multiplier*** in **Lemma** (2) we consider $\theta_n^{(r,k+1)}(t) = Q_n^{(r,k+1)}(t)$ and similarly by **Lemma** (3), we consider $\theta_n^{(r,k)}(t) = Q_n^{(r,k)}(t)$. The only difference from the queue dynamics is that it may not be always possible to transfer packets between queues due to packet availability. Therefore, we replace equality with an inequality,

$$\begin{aligned}{Q_n}^{(r,k)}(t+1) \le \Bigg[Q_n^{(r,k)}(t) + \Bigg[\lambda^{(r)}(t) + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)}(t)\\ + a_n^{(r,k-1)}(t) - \sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)}(t) - a_n^{(r,k)}(t)\Bigg]\Bigg]^+.\end{aligned} \tag{35}$$

To summarize, we formulate the computing and communication resource allocation problem as a ***Lagrangian dual*** problem and utilize the techniques of estimating the ***Lagrange multiplier*** using the queue dynamics, leading to a backpressure-based scheduling and routing algorithm.

APPENDIX E
PROOF OF LEMMA 1

Proof. Fixing a $(r, k) \in (\mathcal{R}, \mathcal{K})$ for each $n \in \mathcal{N}$ in Eq. (24) yields,

$$\begin{aligned}(\Delta Q(t)) &= \frac{1}{2}[Q_n^{(r,k)}(t+1)]^2 - \frac{1}{2}[Q_n^{(r,k)}(t)]^2\\ &= \frac{1}{2}\big([Q_n^{(r,k)}(t)]^2 + 2Q_n^{(r,k)}(t)\cdot h(y_n^{(r,k)}) + [h(y_n^{(r,k)})]^2\big)\\ &- \frac{1}{2}[Q_n^{(r,k)}(t)]^2 = Q_n^{(r,k)}(t)\cdot h(y_n^{(r,k)}) + \frac{1}{2}[h(y_n^{(r,k)})]^2.\end{aligned}$$

We replace the value of $h(y_n^{(r,k)})$ from Eq. (8) and with some rearrangements of the terms we get

$$\begin{aligned}&-Q_n^{(r,k)}(t)\Bigg([\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)} + a_n^{(r,k)}]\\ &\qquad - [\lambda^{(r)} + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)} + a_n^{(r,k-1)}]\Bigg)\\ &+\frac{1}{2}\Bigg([\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)} + a_n^{(r,k)}]^2 + [\lambda^{(r)} + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)} + a_n^{(r,k-1)}]^2\\ &-2[\sum_{j\in\mathcal{N}} s_{n,j}^{(r,k)} + a_n^{(r,k)}][\lambda^{(r)} + \sum_{i\in\mathcal{N}} s_{i,n}^{(r,k)} + a_n^{(r,k-1)}]\Bigg)\end{aligned} \tag{36}$$

Using this definition of G and ϵ, and using the fact that all the variables are non-negative, we simplify Eq.(36) as

$$(\Delta Q(t)) \leq G - \epsilon \cdot Q_n^{(r,k)}(t),$$

which establishes a finite-drift bound and concludes the proof. □

APPENDIX F
PROOF OF NETWORK STABILITY AND NEAR-OPTIMAL UTILITY.

Proof. Assuming that (27) holds, and using the conditional *Lyapunov drift* in (26) we get

$$\mathbb{E}\{L(\overline{\boldsymbol{Q}}(t+1)) - L(\overline{\boldsymbol{Q}}(t)) \mid \overline{\boldsymbol{Q}}(t)\} - V\sum_{r=1}^{R}\mathbb{E}\{U_r(\lambda^{(r)}(t)) \mid \overline{\boldsymbol{Q}}(t)\} \leq G - \epsilon \sum_{\substack{(r,k),\\ n}} Q_n^{(r,k)}(t) - VU(\lambda^*). \quad (37)$$

Taking the expectation over the distribution of the queue backlog process $\overline{\boldsymbol{Q}}(t)$ and using the result of the law of total expectation, i.e., $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X \mid Y))$ yields,

$$\mathbb{E}\{L(\overline{\boldsymbol{Q}}(t+1)) - L(\overline{\boldsymbol{Q}}(t))\} - V\sum_{r=1}^{R}\mathbb{E}\{U_r(\lambda^{(r)}(t))\} \leq G - \epsilon \sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} - VU(\lambda^*). \quad (38)$$

Since the above Lyapunov bound holds for all t, summing over $t \in \{0, 1, ...T-1\}$ gives

$$\mathbb{E}\{L(\overline{\boldsymbol{Q}}(T)) - L(\overline{\boldsymbol{Q}}(0))\} - V\sum_{t=0}^{T-1}\sum_{r=1}^{R}\mathbb{E}\{U_r(\lambda^{(r)}(t))\} \leq GT - \epsilon \sum_{t=0}^{T-1}\sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} - VT\cdot U(\lambda^*). \quad (39)$$

Further, using the non-negativity of the *Lyapunov function* as well as the finite utility bounds, and dividing (39) by $T\cdot\epsilon$, we reach the following result after rearranging the terms.

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(0))\}}{T\cdot\epsilon} \leq \frac{V}{T\cdot\epsilon}\sum_{t=0}^{T-1}U^{max} + \frac{G}{\epsilon} - \frac{V\cdot U(\lambda^*)}{\epsilon} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(T))\}}{T\cdot\epsilon}.$$

Since the right-hand side is the upper bound, we remove the negative terms and complete the sum of maximum utilities, i.e., U^{max}, which is a constant term.

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(0))\}}{T\cdot\epsilon} \leq \frac{VU^{max}+G}{\epsilon}.$$

Finally, we take lim sup as $T\to\infty$ and conclude that,

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{\substack{(r,k),\\ n}} \mathbb{E}\{Q_n^{(r,k)}(t)\} \leq \frac{VU_r^{max}+G}{\epsilon}.$$

The above result shows the stability of queue backlogs which justifies finite queue backlog.

To prove the utility optimality of the proposed algorithm, we first define the optimal utility value ν^*. Assume that, there exists a randomized stationary policy (Ω-only policy) with the control decision λ^* that are deterministically achievable by $\hat{\lambda}_n(t)$. Therefore, this Ω-only policy with λ^* satisfies the following for any $\delta > 0$.

$$\mathbb{E}\{U(\lambda^*)\} + \delta \geq \nu^*. \quad (40)$$

Intuitively, satisfying all the constraints λ^* produces a utility value at least as good as ν^*. For convenience, assume $\delta = 0$ and replace the inequality with equality (the same result can be achieved by taking $\lim\delta\to 0$) [42]. Finally, rearranging the terms after dividing (39) by $T\cdot V$ and plugging in Eq. (40) yield the following.

$$\sum_{r=1}^{R}\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\{U_r(\lambda^{(r)}(t))\} \geq \nu^* + \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(T))\}}{T\cdot V} - \frac{G}{V} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(0))\}}{T\cdot V} + \frac{\epsilon\sum_{t=0}^{T-1}\sum_{(r,k),n}\mathbb{E}\{Q_n^{(r,k)}(t)\}}{T\cdot V} \geq \nu^* - \frac{G}{V} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(0))\}}{T\cdot V}. \quad (41)$$

Due to Assumption 2 (Infinite reservoir and real-value assumptions) and concavity of the utility function, *Jensen inequality* holds. Therefore, we get

$$\sum_{r=1}^{R}U_r(\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{\lambda^{(r)}(t)\}) \geq \nu^* - \frac{G}{V} - \frac{\mathbb{E}\{L(\overline{\boldsymbol{Q}}(0))\}}{T\cdot V}. \quad (42)$$

By taking the lim inf as $T\to\infty$ and the result from Assumption 1, we conclude

$$\liminf_{T\to\infty}\sum_{r=1}^{R}U_r(\overline{\lambda}^{(r)}(T)) \geq \nu^* - \frac{G}{V}.$$

□

APPENDIX G
DISTRIBUTED IMPLEMENTATION

Here, it must be stressed that even though the flow rate controller in **(SP1)** and internal forwarding in **(SP2A)** are solved in a fully distributed manner, due to the interference in the wireless access media the external forwarding in **(SP2B)** is assumed to be centralized. The max-weight solution for the external forwarding problem uses the local queue length information in the neighboring nodes. However, the max weight matching is difficult to compute in a distributed manner since matching constraints $\sum_{n,j\in\mathcal{L}}\mu_{n,j}\cdot w_{n,j}$ couple the network-wide decision. The result of our work can be implemented using a simpler interference model. For example, in [43] a general interference model is considered where a node has the full knowledge of contenders of link (n,j) denoted by $\Psi_{n,j}$ that cannot be activated when link (n,j) transmits. Define $\Psi_{max} \triangleq max_{n,j}\ \Psi_{n,j}$. Assume a simple random access scheme that uses this collision model. The model requires only one contention round to find non-interfering links in the following manner. Each link independently attempts the activation with probability $\frac{1}{\Psi_{max}}$. If there is no collision, the link is activated and remains idle otherwise. The random access scheme is similar to the distributed approach [39]. In our work, we do not manage the random access but assume a computing node is given a set of collision-free communication links to communicate with the neighboring computing nodes. Under such an assumption, the computing nodes distributedly schedule the external packet forwarding.

KM Mahfujul is currently working toward his Ph.D. degree in Electrical and Computer Engineering at Queen's University, ON, Canada. He received his M.Sc. degree in computer science and technology from Central South University, Hunan, China, in 2020 and his B.Sc. degree in computer science and engineering from Khulna University, Khulna, Bangladesh in 2016. His current research interests include real-time scheduling and distributed algorithm design for converged wireless/cloud communication networks.

Kaige Qu (S'19−M'21) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2021. She received the B.Sc. degree in communication engineering from Shandong University, Jinan, China, in 2013, and M.Sc. degrees in integrated circuits engineering and electrical engineering from Tsinghua University, Beijing, China, and KU Leuven, Leuven, Belgium, respectively, in 2016. From February 2021 to December 2023, she was a Post-doctoral Fellow and then a Research Associate with the Department of Electrical and Computer Engineering, University of Waterloo. Her research interests include connected and autonomous vehicles, network intelligence, network virtualization, and digital twin assisted network automation.

Dr. Qiang (John) Ye received the PhD degree in Electrical and Computer Engineering from the University of Waterloo, ON, Canada, in 2016. Since Sept. 2023, he has been an Assistant Professor with the Department of Electrical and Software Engineering, Schulich School of Engineering, University of Calgary, AB, Canada. Before joining UCalgary, he worked as an Assistant Professor with the Department of Computer Science, Memorial University of Newfoundland, NL, Canada from Sept. 2021 to Aug. 2023 and with the Department of Electrical and Computer Engineering and Technology, Minnesota State University, Mankato, USA, from Sept. 2019 to Aug. 2021, respectively. He was with the Department of Electrical and Computer Engineering, University of Waterloo as a Postdoctoral Fellow and then a Research Associate from Dec. 2016 to Sept. 2019.

He has published over 70 research articles on top-ranked journals and conference proceedings. He is/was the General, Publication, Program Co-chairs for different reputable international conferences and workshops, and serves/served as Associate Editors of prestigious international journals, e.g., IEEE Transactions on Vehicular Technology and IEEE Transactions on Cognitive Communications and Networking. He also serves/served as the IEEE Vehicular Technology Society (VTS) Region 7 Chapter Coordinator (2024) and the Regions 1-7 Chapters Coordinator(2022-2023). Dr. Ye received the Best Paper Award in the IEEE/CIC International Conference on Communications in China (ICCC) in 2024 and the IEEE Transactions on Cognitive Communications and Networking Exemplary Editor Award in 2023. He is a Senior Member of IEEE

Ning Lu (Member, IEEE) received the B.Eng. and M.Eng. degrees in electrical engineering from Tongji University, Shanghai, China, in 2007 and 2010, respectively, and the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering with Queen's University, Kingston, ON, Canada. Prior to joining Queen's University, he was an Assistant Professor with the Department of Computing Science, Thompson Rivers University, Kamloops, BC, Canada. From 2015 to 2016, he was a Postdoctoral Fellow with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Champaign, IL, USA. He was as an Intern with the National Institute of Informatics, Tokyo, Japan, in Summer 2009.

He has published more than 70 papers in top IEEE journals and conferences, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, ACM MobiHoc, and IEEE INFOCOM. His current research interests include scheduling, distributed algorithms, and reinforcement learning for wireless communication networks. Dr. Lu is currently an Editor of the IEEE Transactions on Wireless Communications and chairing the Special Interest Group (SIG) on AI Empowered Internet of Vehicles (IoV), IEEE Cognitive Networks Technical Committee.