Robotic Message Ferrying for Wireless Networks using Coarse-Grained Backpressure Control

Shangxing Wang, Andrea Gasparri, *Member, IEEE,* and Bhaskar Krishnamachari, *Member, IEEE*

Abstract—We formulate the problem of robots ferrying messages between statically-placed source and sink pairs that they can communicate with wirelessly. We first analyze the capacity region for this problem under ideal conditions. We indicate how robots could be scheduled optimally to satisfy any arrival rate in the capacity region, given prior knowledge about arrival rate. We then consider the setting where the arrival rate is unknown and present a coarse-grained backpressure message ferrying algorithm (CBMF) for it. In CBMF, the robots are matched to sources and sinks once every epoch to maximize a queue-differential-based weight. The matching controls both motion and transmission for each robot. We show through analysis and simulations the conditions under which CBMF can stabilize the network, and its corresponding delay performance. From a practical point of view, we propose a heuristic approach to adapt the epoch duration according to network conditions that can improve the end-to-end delay while guaranteeing the network stability at the same time. We also study the structural properties with its explicit delay performance of the CBMF algorithm in a homogeneous network.

Index Terms—Message ferrying, Robotic wireless networks, Backpressure, Adaptive algorithm

1 Introduction

Since the work by Tse and Grossglauser [1], it has been known that the use of delay tolerant mobile communications can dramatically increase the capacity of wireless networks by providing ideal constant throughput scaling with network size at the expense of delay. However, though the idea of message ferrying using controllable mobility nodes dates back to the work by Zhao and Ammar [2], nearly all the work to date has focused on message ferrying in intermittently connected mobile networks where the mobility is either unpredictable, or predictable but uncontrollable. With the rapidly growing interest in multi-robot systems, we are entering an era where the position of network elements can be explicitly controlled in order to improve communication performance.

This paper explores the fundamental limits of robotically controlled message ferrying in a wireless network. We consider a setting in which a set of K pairs of static wireless nodes act as sources and sinks that communicate not directly with each other (possibly because they are located far from each other and hence cannot communicate with each other at sufficiently high rates) but through a set of N controllable robots. We assume there is a centralized control plane responsible for scheduling robots. Because it collects only queue state information about all network entities, this centralized plan can be relatively inexpensively created either using infrastructure such as cellular / WiFi, or through a low-rate multi-hopping mesh overlay.

We mathematically characterize the capacity region of this system, considering ideal (arbitrarily large) settings with respect to robot mobility and scheduling durations. Our analysis shows that with $N = 2K$ robots the system could be made to operate at full capacity (arbitrarily close to the throughput that could be achieved if all sources and sinks were adjacent to each other). We indicate how any traffic that is within the capacity region of this network can be served stably if the data arrival rate is known to the scheduler. We then consider how to schedule the robots when the arrival rate is not known *a priori*. For this case, we propose and evaluate a queue-backpressure based algorithm for message ferrying that is coarse-grained in the sense that robot motion and relaying decisions are made once every fixed-duration epoch. We show that as the epoch duration and velocity of robots both increase, the throughput performance of this algorithm rapidly approaches that of the ideal case. In addition, to improve the performance of the CBMF algorithm in practice, we design a heuristic scheme to show how one can adapt network settings according to network conditions to reduce delay with a guaranteed throughput performance. What's more, to gain insights on its implementation in practice, we also conduct a study on the structural properties and corresponding performance of the CBMF algorithm in a homogeneous network where all source-sink pairs have same data arrival rate and same delivery distance.

In summary, the major contributions of this paper are

- Developing a framework to mathematically characterize the capacity and delay performance of a robotic wireless network.

- *S. Wang and B. Krishnamachari are with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, 90007.*
  E-mail: shangxiw@usc.edu and bkrishna@usc.edu
- *A. Gasparri is with the Department of Engineering, Roma Tre University, Rome, Italy.*
  E-mail: gasparri@dia.uniroma3.it

- Proposing the Coarse-Grained Backpressure-based Message Ferrying (CBMF) algorithm that works for unknown arrival rate and studying its corresponding capacity and delay performance.
- Designing a heuristic approach to adapt network settings to keep the network stable while maintaining small delay.
- Studying the structural properties and explicit delay performance of the CBMF algorithm in a multi-flow homogeneous network.

The rest of the paper is organized as follows: In Section 2, the related work is reviewed; In Section 3 we describe our problem formulation; in Section 4 we present how to schedule robots if the arrival rate is known, while in Section 5 we propose the CBMF for unknown arrival rate and study its capacity and delay performance; In Section 6 we propose an heuristic approach to adapt the epoch duration; The structural properties of CBMF algorithm are studied in Section 7; In Section 8, numerical results are presented to validate our analysis and we conclude this paper in Section 9.

2 RELATED WORK

In the mobile networking community, controlled mobility has become a new design dimension to improve the network performance. In [2], Zhao *et al.* introduces the concept of *message ferries*, which are mobile devices that can proactively move around by following their pre-designed routes to help deliver data. Based on the idea of non-random movement, they propose a proactive routing scheme for a single ferry to provide regular connectivity for disconnected static wireless ad hoc networks. Later works start to consider more general cases, such as multi-ferry control [3] and delay tolerant networks [4]. Meanwhile in [5], Goldenberg *et al.* consider mobility as a network control primitive, and present the first distributed and self-adaptive mobility control scheme for improving communication performance.

On the other hand, controlled mobility is always an active research area in the robotics community. In coordinated robots systems, mobility controllers of agents can be designed through local interactions to allow robots to perform useful collective behaviors such as flocking [6], formation control [7] and swarming [8]. Though communication plays a critical role in the coordinated systems, none of these works consider practical communication models but only a simple disc model, which does not fit in a realistic environment and thus degrades the performance.

In recent years, with the advent of the integration of mobile robotics and wireless networking, realistic communication factors together with routing issues have become an emerging research topic. Researchers in [9] have studied an integrity problem where controllers of robots are designed to conduct some task while maintaining certain desired end-to-end transmission rates at the same time. A robotic router formulation problem has been studied in [10], where an optimal configuration of robots is formulated to maintain a maximized successful reception rate in realistic communication environments that naturally experience path loss, shadowing, and multipath fading. The expected number of transmissions per successfully delivered packet (ETX) has also been taken into consideration in [11], where researchers design a hybrid architecture to allow robots optimally configured so that each flow has a minimized ETX in a multiple flow network. All of these works consider the joint robotic control and transmission scheduling based on a convex optimization approach, while our work designs schemes from a queue-awareness perspective.

Inspirations for this work came from the backepressure scheduling and routing in wireless networks. In [12], Tassiulas and Ephremides propose the original idea of backpressure-based queue weight maximization scheduling scheme, and show that it can guarantee the stability of a general network for any arrival rate within an optimal capacity region. The essence of backpressure scheduling is to prioritize transmissions over links that have the highest queue differentials. This mechanism is, to some extend, similar to a gradient descent approach where the gradient refers to queue differential. Later, researchers incorporate the original results with utility optimization and presents backpressure scheduling is a throughput-optimal and simple network protocol that can integrate medium access, routing, rate and power control for all kinds of networks [14]- [19]. However, large queue sizes have to be maintained when applying backpressure scheduling, which can cause inevitably long delay in the network. Techniques and methods to improve its delay performance have been considered in both theory and practice [21]- [27]. In this work, we combine the idea of backpressure scheduling with robotic control to jointly control the movement and routing in the robotic wireless network. Not only a certain capacity region can be guaranteed, the delay performance can also be controlled via a tunable parameter.

Closely related to this work are our two recent conference publications [28], [29]. In [28], we initially propose the idea of queue-aware joint robotic control and transmission scheduling, and design the CBMF algorithm. Built upon our initial work [28], we in this work conduct complete capacity and delay performance analysis of the CBMF algorithm and find a heuristic approach to adapt network settings according to network conditions with the purpose to reduce the network delay while maintaining the network stability at the same time. In [29], we propose a fine-grained backpressure message ferrying algorithm (FBMF) where robots' allocation decision is made in a much finer manner by also taking current transmission rates into consideration. It shows that the FBMF algorithm is throughput optimal for the simplest setting, which is single-flow, single-robot, deterministic arrival. But there is no certain answer about its performance in a general case. In contrast to [29], we in this work study the robotic message ferrying problem with multiple flows and multiple robots in a general scenario, and propose a CBMF algorithm whose throughput performance approaches the ideal capacity region in the general case. Furthermore, delay performance is also studied in this work to gain more understanding.

3 PROBLEM FORMULATION

We consider a network where there are K pairs of static source and destination nodes located at arbitrary locations in a a two- or three-dimensional Euclidean space. Let the

source for the i^{th} flow be denoted as $src(i)$, and the destination or sink for that flow be denoted as $sink(i)$. Packets arrive at a source following either a deterministic process or an i.i.d. stochastic process, and the arrival rate at source i is a constant λ_i[1]. A list of notations is presented in Table 1.

There are N mobile robotic nodes in the same space that act as message ferries, i.e. when they talk to a source node, they can collect packets from it, and when they talk to a sink node, they can transmit packets to it. These robotic message ferries are special helper nodes whose mobility can be controlled to assist communication and enhance the connectivity in a wireless networks. For simplicity, we assume that the static nodes do not communicate directly with each other, but rather only through the mobile robots. Also we, following most previous works ([3], [4]) as well as considering the fact that current hardware cannot support one node simultaneously talking to multiple other notes, assume that a static source or sink node can talk to at most one robot at any time, which indicates at most $2K$ robots are needed. Thus, in the following, we assume $N \leq 2K$.

Time is divided into discrete time steps of unit duration, and every T time steps there is a new epoch. At the start of each epoch, a centralized scheduler can collect useful information from source and sink nodes as well as mobile robots, and use this information to allocate each robot to either a source or sink. The matching is represented by an allocation matrix A such that $A(i,j)$ is 0 if the robot j is not allocated to either source or sink for flow i, 1 if it is allocated to $src(i)$, and -1 if it is allocated to $sink(i)$. When a robot is allocated to a given source (or sink), for the rest of that epoch it moves with a uniform velocity v along the straight line directly to the assigned node until it reaches its position. At all time steps of that epoch that robot will communicate continuously and exclusively with that source (or sink) to collect (or deliver, in case of the sink) any available packets at a rate depending on its current distance to that node. Moreover, orthogonal channels are assigned to communication pairs to avoid interference.

The locations of the sources and sinks for flow i are denoted by $x_{src(i)}$ and $x_{sink(i)}$ respectively, and the location of robot j at time t is denoted as $x_j(t)$. All locations are in the space $\mathbb{R}^n$, where $n \in \{2,3\}$. Let the distance between a source for flow i and a robot j be denoted as $d(x_{src(i)}, x_j(t))$ (similarly for the sink), which is a metric in $\mathbb{R}^n$ (for instance, Euclidean distance). So if robot j is moving towards the source for flow i (similarly for the sink), its position $x_j(t)$ is updated so that it moves along the vector between its previous position and the source location to be at the following distance:

$$d(x_{src(i)}, x_j(t+1)) = max\{d(x_{src(i)}, x_j(t)) - v, 0\} \quad (1)$$

We assume that the rate at which a source for flow i can transmit to a robot j, denoted by $R_{src(i),j}(t)$ is always strictly positive, and decreases monotonically with the distance between them, and similarly for the rate at which a robot j can transmit to the sink for flow i, denoted by $R_{j,sink(i)}(t)$. We assume that when the robot is at a location of a particular source or sink, (i.e., the distance between them is 0), the corresponding throughput between the mobile robot and that source or sink is maximized as R_{max}

In the network, static nodes (sources and sinks) and mobile robots have buffers with infinite size, and any undelivered packet can be stored in the corresponding buffer[2]. The queue at the source for flow i is denoted as $Q_{src(i)}$. It is assumed that there is no queue at the sinks as they directly consume all packets intended for them. Each robot j maintains a separate queue for each flow i, labelled Q_j^i. Figure 1 shows an illustration of this system with $K = 2$ flows and $N = 4$ robots.

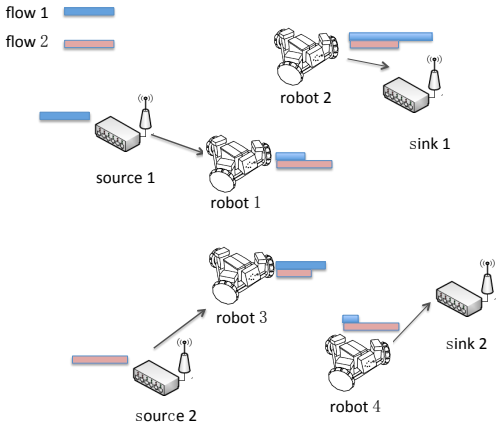

Fig. 1. A network containing 2 pairs of source and sink nodes and 4 robots

TABLE 1
List of main notations in the problem formulation

Parameter	Definition
K	number of source and destination pairs
N	number of mobile robots
$src(i)$	source of the i^{th} flow
$sink(i)$	sink of the i^{th} flow
$x_{src(i)}$	location of the source of the i^{th} flow
$x_{sink(i)}$	location of the sink of the i^{th} flow
$x_j(t)$	location of robot j at time t
$R_{src(i),j}(t)$	transmission rate between a source for flow i and robot j at time t
$R_{j,sink(i)}(t)$	transmission rate between a sink for flow i and robot j at time t
$Q_{src(i)}(t)$	queue size at the source for flow i at time t
$Q_j^i(t)$	queue size at robot j for flow i at time t
λ_i	packet arrival rate at the source of flow i
T	epoch length
v	robot's velocity
A	allocation matrix

1. If the data arrival process at source i is stochastic, λ_i is the expectation. Further, we assume the second moment of the stochastic process is finite.

2. In theory, no limit on the number of messages a robot can carry is the ideal case, and this assumption enables us to have a clear mathematical treatment of the capacity and delay performance study in later sections. In practice, the maximum buffer occupancy will show a concentration around some nominal value which depends upon how close the arrival rate gets to the boundary of the capacity region; this can be used to determine how to size the buffers to be a finite value while ensuring a negligible packet drop probability.

Therefore, if a robot j is communicating with $src(i)$ at time t, the update equations for the corresponding queue of the robot and the source queue will be as follows:

$$\begin{aligned} n_p(t) &= \min\{R_{src(i),j}(t), Q_{src(i)}(t)\} \\ Q_j^i(t+1) &= Q_j^i(t) + n_p(t) \\ Q_{src(i)}(t+1) &= Q_{src(i)}(t) - n_p(t) + \lambda_i \end{aligned} \tag{2}$$

Similarly, if the robot j is communicating with $sink(i)$ at time t, the queue update equation for the robot's corresponding queue will be:

$$\begin{aligned} n_q(t) &= \min\{R_{j,sink(i)}(t), Q_j^i(t)\} \\ Q_j^i(t+1) &= Q_j^i(t) - n_q(t) \end{aligned} \tag{3}$$

The above formulated system model will be used in all following sections. And the goal of this work is to study and design scheduling and allocation algorithms about how the centralized scheduler allocates mobile robots to improve the communication performance of the robotic message ferrying network.

4 Capacity Analysis

In this section, we aim at finding the capacity region of the network of robotic message ferrying problem under the following assumptions: a) The message arrival rates at sources are known; b) The epoch length T and robots' velocity v can be set arbitrarily large. In other words, we want to find the largest rate region such that any arrival rate vector[3] within this region can be stably served (i.e., the average size of each queue can be maintained to be bounded) under ideal conditions. This capacity region serves as a performance upper bound that will guide us to design robotic scheduling algorithms and evaluate corresponding performances under practical conditions in later sections.

Definition 1. *(Capacity Region) The capacity region is the set of all arrival rate vectors that are stably supportable by the network, considering all possible scheduling policies.*

Based on the previous system model of the robotic message ferrying problem, we are able to find its capacity region:

Theorem 1. *The capacity region of the robotic message ferrying problem is an open region $\mathbf{\Lambda}$ of arrival rate vectors as follows*

$$\mathbf{\Lambda} = \left\{ \lambda | 0 \le \lambda_i < R_{\max}, \quad \forall\, i,\ \sum_{i=1}^{K} \lambda_i < \frac{R_{\max} N}{2} \right\} \tag{4}$$

Proof. See proof in Appendix A. □

The general idea of the proof is to show that this arrival rate region $\mathbf{\Lambda}$ can be served by a convex combination of "basis" configurations in which robots are allocated to serve distinct flows. Let $\tilde{\Gamma}$ be a finite set of vectors defined as:

$$\tilde{\Gamma} = \left\{ \gamma | \gamma_i = \frac{a_i R_{\max}}{2}, \quad \forall\, i, a_i \in \{0,1,2\}, \sum_{i=1}^{K} a_i \le N \right\} \tag{5}$$

Each element of this set $\tilde{\Gamma}$ is an arrival rate vector of one basis configuration, and the corresponding integer vector $\mathbf{a}$ corresponds to a basis allocation of robots to distinct sources and sinks that can serve each flow at rate γ_i. Specifically, a_i refers to the number of robots allocated to serve flow i. And each basis allocation corresponding to the elements of $\tilde{\Gamma}$ can actually be expressed as two distinct but symmetric allocations of robots to sources/sinks over two successive epochs. For the i^{th} flow, if $a_i = 0$, there is no robot allocated to either the source or sink in either of these two epochs, yielding a service rate of $\gamma_i = 0$; if $a_i = 1$, a particular robot is assigned to be at the source at the first epoch and at the sink at the second epoch, yielding a service rate of $\gamma_i = \frac{R_{max}}{2}$; if $a_i = 2$, two robots are assigned (call them R_1 and R_2) such that R_1 is at the source at the first epoch and at the sink at the second epoch while R_2 is at the sink at the first epoch and at the source at the second epoch, yielding a service rate of $\gamma_i = R_{max}$. The constraints on a_i ensure that the total number of robots allocated does not exceed the available number N.

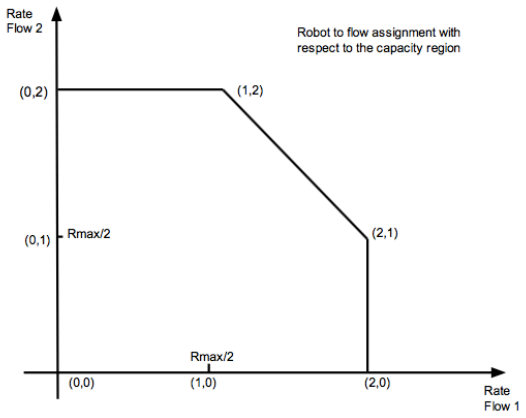

Fig. 2. Capacity region for a problem with 3 robots and 2 flows

Figure 2 shows an example of the capacity region when the robotic message ferrying network has two source-sink flows and three mobile robots, i.e., $K = 2$ and $N = 3$. The labels such as (x, y) are given to the basis allocations on the Pareto boundary to denote that they can be achieved by allocating an integer number of robots x to flow 1 and y to flow 2. Note in particular that the point (R_{max}, R_{max}) is outside the region in this case because the only way to serve that rate is to allocate two robots full time to each of the two flows, and we have only 3 robots. The vertices on the boundary of the region, which represent basis allocations, are all in the set $\tilde{\Gamma}$; the convex hull of $\tilde{\Gamma}$ completely describes the region.

5 Coarse-Grained Backpressure Control

From the previous discussion, we know that if the arrival rate is known, and within the ideal capacity region of the system, a service schedule for the robots can be designed in such a way that the arrival rate is served in a stable manner. The analysis thus far assumes that either the velocity of the robot or the epoch duration can be chosen to be arbitrarily large, which may not be true in practice. In the following, motivated by practical considerations we consider the case

3. An arrival rate vector is a vector of arrival rates of flows.

when T and v are finite and fixed, and study the corresponding capacity region.

5.1 Capacity Region under finite velocity and epoch duration

Assume the epoch length T and the velocity v are finite and fixed. In particular, the restriction of T to be finite is useful for two reasons: a) it fixes the overhead of scheduling and b) it can be used to enforce an upper bound on delay (time taken for a packet generated at the source to reach the sink). As may be expected, these constraints reduce the capacity region.

The fraction of time spent in transit, is bounded by $\frac{d_{max}}{vT}$. We assume that $\frac{d_{max}}{vT} < 1$, which implies that a robot can always reach its assigned node (source or sink) within an epoch. Then the average transmission rate R_{avg} during an epoch in the worst case a robot can achieve can be derived as follows. Consider the worst case where at the beginning of an epoch a robot is allocated to collect data from a source that is at the longest distance of d_{max} away[4]. The data transmission rate of a robot at time t is $R(d_{max} - vt)$, and it will take the robot $\frac{d_{max}}{v}$ time to reach the assigned node. When the robot reaches the node, it stays there and maintains the maximum transmission rate R_{max} for the rest of an epoch.

Therefore, the total number of data that can be collected by the robot during an epoch consists of two parts: the first part is the maximum number of data that can be collected during a robot's movement, which equals $\int_0^{\frac{d}{v}} R(d_{max} - vt)dt$, and the second part is the maximum number of data that can be collected when the robot stays at its assigned node, which is $R_{max}(T - \frac{d}{v})$. Thus, the average transmission rate in an epoch is:

$$R_{avg} = \frac{1}{T}\left\{ \int_0^{\frac{d}{v}} R(d_{max} - vt)dt + R_{max}(T - \frac{d}{v}) \right\} \quad (6)$$

This directly provides an inner-bound on the capacity region for finite v and T expressed in terms of R_{avg}, which can still be achieved while scheduling robots in the same way by a convex combination of configurations in which robots are allocated to serve distinct flows as that in Section IV:

$$\mathbf{\Lambda_{IB}}(\mathbf{v}, \mathbf{T}) = \left\{ \lambda | 0 \leq \lambda_i < R_{avg},\ \forall\, i,\ \sum_{i=1}^{K} \lambda_i < \frac{R_{avg} N}{2} \right\} \quad (7)$$

5.2 Coarse-grained Backpressure-based Message Ferrying

In previous discussions, for any packets' arrival rate in the capacity region, as long as they are known *a priori*, they can be stably served. We now consider a more practical case when the arrival rate is not known to the centralized scheduler, while the the epoch duration T and robots' velocity v are still kept finite and fixed. Is it still possible to schedule the movements and communications of robots in such a way that all queues remain stable? And what is the corresponding rate region that can be achieved?

The answer to this question turns out to be yes, using the notion of Backpressure scheduling first proposed by Tassiulas and Ephremides [12]. We propose an algorithm for scheduling message ferrying robots only based on queue information for finite v and T parameters, which we refer to as coarse-grained backpressure-based message ferrying (CBMF) presented in Algorithm 1.

Algorithm 1 CBMF Algorithm

for $n = 1, 2, \ldots$ **do**
  At the beginning of epoch n
  for $i = 1, \ldots, K$ **do**
    for $j = 1, \ldots, N$ **do**
      $w_{src(i),j} = Q_{src(i)}(nT) - Q^i_j(nT)$
      $w_{sink(i),j} = Q^i_j(nT)$
    end for
  end for
  Find an allocation matrix A that solves optimization problem (P1)
end for

The Optimization problem (P1) is formulated as follows:

$$\begin{aligned} \underset{A}{\text{maximize}} \quad & \sum_{i,j} w(A(i,j)) \\ \text{subject to} \quad & \sum_{i} |A(i,j)| = 1,\ \forall j \qquad (a) \\ & \sum_{j} \mathcal{I}\{A(i,j) = 1\} \leq 1,\ \forall i \qquad (b) \\ & \sum_{j} \mathcal{I}\{A(i,j) = -1\} \leq 1,\ \forall i \qquad (c) \end{aligned} \quad \text{(P1)}$$

where $w : \mathcal{X} = \{-1, 0, 1\} \rightarrow \mathcal{R}$ is a function defined as follows:

$$w(x) = \begin{cases} w_{src(i),j} & \text{if } x = 1 \\ w_{sink(i),j} & \text{if } x = -1 \\ 0 & \text{if } x = 0 \end{cases} \quad (8)$$

The constraint (a) in (P1) ensures that each robot is allocated to exactly one source or sink. The constraint (b) in (P1) ($\mathcal{I}\{\}$ represents the indicator function) ensures that no source is allocated more than one robot, while the constraint (c) in (P1) ensures that no sink is allocated more than one robot.

The corresponding capacity and delay performance of the CBMF algorithm is shown in Theorem 2.

Theorem 2. *For any arrival rate that is within $\mathbf{\Lambda_{IB}}(\mathbf{v}, \mathbf{T})$, the CBMF algorithm ensures that all source and robot queues are stable (always bounded by a finite value).*

Proof. See proof in Appendix B. □

The proof of this theorem follows from bounding the drift of a quadratic Lyapunov function and deriving a control policy that minimizes this bound, following closely the approach pioneered by Tassiulas and Ephremides [12]. The technical complication in this setting compared to traditional backpressure as applied to static wireless networks is that the average rate obtained over the course of an epoch

4. Though R_{avg} is analyzed in the context of robot collecting data from a source, the same result can be derived if we focus on an epoch where a robot delivers data to a sink.

for each matching can be slightly different depending on the starting position of the robot with respect to the node it is being matched to. CBMF treats the rate for each matching to be the same as R_{avg} in its weight calculation, and as a result it is not provably stable for all arrival rate vectors in the ideal capacity region (which as discussed before is in fact achievable using scheduling with prior knowledge of the arrival rates); this theoretical guarantee can only be provided for all arrival rates up to the inner-bound. However, as v and T increase, the inner-bound approaches the ideal capacity region.

Remark 1. *In the proof in Appendix B, we also show that for a fixed arrival rate, the end-to-end delay scales as $O(T)$ so long as T is large enough to ensure that the system remains stable.*

Also, we would like to point out that the optimization problem (P1) in Algorithm 1 is actually a Max-Weighted Bipartite Matching problem, where the set of robots and the set of static nodes (i.e., sources and sinks) form a bipartite graph with edges connecting each robot and each static node with edge weights as $w_{src(i),j}$ and $w_{sink(i),j}$ accordingly. The optimal allocation can be computed in polynomial time by using some well-known algorithms, such as Hungarian algorithm [13]. This makes the CBMF algorithm works efficiently in practice.

6 Epoch Adaptive CBMF

According to Remark 1 we know that when the number of flows and the number of robots are fixed, the end-to-end delay increases linearly as the epoch length T increases when the network is stable under the CBMF algorithm. Therefore, reducing epoch length can help reduce network delay as long as the network is maintained stable. If the arrival rates of flows in the network are given, theoretically, we could apply eqns. (6) and (7) to find the best epoch length T for the CBMF algorithm to make the network stable while maintaining a small (if not the smallest) delay. But this requires us to know all related network settings including the number of flows and robots, locations of sources and sinks, velocity of robots and communication model and its corresponding parameters, not all of which could be available *a priori* in practice. Moreover, the arrival rate of flows is probably unknown and may change during run time as well.

Therefore, we propose a heuristic approach to show how one can adapt the epoch length T in the implementation of the CBMF algorithm to make a network stable according to current network condition while maintaining a small delay without necessity of knowing all information, as in Algorithm 2.

The general idea of Algorithm 2 is to initially set T as a very large positive constant T_{th} to make sure the network is stable for any arrival rate. The centralized coordinator allocates robots according to Algorithm 1, and reduces the epoch duration T iteratively by a small step size δ every L observation duration until the network delay cannot be decreased. Then the robots are allocated according to Algorithm 1 for the fixed finalized T and a small network delay can be maintained. The algorithm works well when the arrival rate λ remains constant. However, the system becomes unstable when arrival rates increase. To solve this problem, one can keep observing the network delay while applying Algorithm 2. And when the network delay increases, which indicates the arrival rate vector is beyond the capacity region for the current epoch duration T, one can reset the epoch duration back to the maximum value T_{th} and re-run Algorithm 2.

Algorithm 2 Epoch Adaptive CBMF Algorithm

```
Initially set T = T_th and D_old = D_new = ∞.
while D_old ≥ D_new do
    D_old ← D_new
    T ← T − δ                    ▷ Skip this step in the initial loop
    Run Algorithm 1 for a duration L        ▷ L >> T
    Set D_new as current network delay
end while
T ← T + δ
Run Algorithm 1
```

7 Structural Properties and Delay Performance of CBMF Algorithm in a Homogeneous Network

It has been shown in Theorem 2 that the CBMF algorithm can provide the stability of a network for any arrival rate vector within the capacity region $\mathbf{\Lambda_{IB}}(\mathbf{v}, \mathbf{T})$. And the essence is to find an allocation of robots that solves the optimization problem (P1) at the beginning of each epoch, which also indicates the total queue differential is maximized. However, this does not provide us with any clue about the structures or patterns of allocations. In this section, we study a homogeneous network with deterministic data arrival process and aim at finding structural properties of the CBMF that can be utilized in practice to improve performance.

7.1 Structural Properties

We consider a homogeneous network in the sense that every flow has the same arrival rate and the distance between a source, and its corresponding sink is the same among all K flows. Initially, we assume all queues are empty.

In the following discussion, in the case when the number of robots assigned to collect data from sources is no greater than the number of flows in a network, we assume that a collecting robot can collect all data in the source and the source queue become empty at the end of the epoch. This assumption is reasonable and reflects the actual performance most of the time if the arrival rate vector is within the capacity inner bound. Also if there are data keeping accumulated at the source, the network can no longer remain stable.

As observed, when applying the CBMF algorithm to find robots' allocation for each epoch, there may be multiple allocations available, all of which are solutions to the optimization problem (P1). Thus, we make two preferences when choosing one allocation out of multiple possible allocations: a) based on the intuition that better delay performance can be achieved if robots can deliver data to corresponding sinks as soon as possible, we let the centralized coordinator prefer choosing the allocation which can assign as many robots as possible to sinks to deliver data whenever there

is queue differential existing between a robot and a sink; b) if no queue differential exists between any robot and sink, the centralized coordinator prefers choosing the allocation which can assign as many robots as possible to sources to collect data.

Remark 2. *These two preferences only deal with the case when there are multiple possible allocations that solves the optimization problem* (P1)*. Therefore, the robotic allocation from CBMF algorithm with these two preferences can still stabilize the network for any arrival rate vector within the capacity inner bound* $\mathbf{\Lambda_{IB}}(\mathbf{v}, \mathbf{T})$.

With the above two preferences, we are able to study the structural properties of the CBMF algorithm in the multi-flow homogeneous network and find the robotic allocation from the CBMF algorithm has a time sharing structure. In the following, we present the structural property result in two different cases depending on the number of flows and the number of robots. First, for the case when the number of robots is no more than the number of flows, we have the following result:

Lemma 1. *In a multi-flow homogeneous network, when the number of robots is no greater than the number of flows, i.e.,* $N \leq K$*, the CBMF Algrotihm with an allocation that exactly solves the optimization problem* (P1) *at the start of each epoch is equivalent to the Robot Allocation Strategy I shown in Algorithm 3.*

Proof. See proof in Appendix C. □

Algorithm 3 Robot Allocation Strategy I

1: At the beginning of the initial epoch, randomly pick N sources and allocate a robot to each chosen source.
2: **for** $n = 2, 3, \ldots$ **do**
3: At the beginning of epoch n
4: **if** n is odd **then**
5: Allocate each robot to one of the N least recent served sources to collect data
6: **else**
7: Allocate each robot to a sink corresponding to its previously assigned source to deliver data
8: **end if**
9: **end for**

For the case when the number of robots is greater than the number of flows, the corresponding result is shown as follows:

Lemma 2. *In a multi-flow homogeneous network, when the number of robots is greater than the number of flows, i.e.,* $K < N \leq 2K$*, the CBMF algorithm with an allocation that exactly solves the optimization problem* (P1) *at the start of each epoch is equivalent to the Robot Allocation Strategy II shown in Algorithm 4.*

Proof. See proof in Appendix D. □

Therefore, in a homogeneous network, the CBMF algorithm has a simple time sharing structure (either Algorithm 3 or 4). The centralized coordinator does not need to collect any queue information for making an allocation decision. It only requires the centralized coordinator to find out the least recent served flows, which can be done by keeping a record in memory. In addition, every robot can have a finite buffer since it serves a source and its corresponding sink in two consecutive epochs. All of these can make our algorithms easy and efficient to implement in practice

Algorithm 4 Robot Allocation Strategy II

1: Divide N robots into two groups. Group I contains K robots and Group II contains $N - K$ robots.
2: At the beginning of the initial epoch, randomly allocate each robot in Group I to a source; And randomly choose $N - K$ sinks and allocate each robot in Group II to a chosen sink
3: **for** $n = 2, 3, \ldots$ **do**
4: At the beginning of epoch n
5: **if** n is odd **then**
6: Allocate each robot in Group I to a source to collect data
7: Allocate each robot in Group II to a sink corresponding to its previously assigned source to deliver data
8: **else**
9: Allocate each robot in Group I to a sink corresponding to its previously assigned source to deliver data
10: Allocate each robot in Group II to one of the $N - K$ least recent served sources to collect data
11: **end if**
12: **end for**

7.2 Delay Analysis

According to Remark 1, the end-to-end delay scales as $O(T)$ so long as T is large enough to make sure the system is stable for a fixed arrival rate. However, this scaling law perspective still cannot provide a clear idea of how delay performs when the epoch length is finite. In a multi-flow homogeneous network, the CBMF algorithm becomes as simple as a time-sharing robot allocation strategy (either Algorithm 3 or 4), which enables us to find its explicit delay performance:

Theorem 3. *In a homogeneous network with* K *flows and* N *robots, the network delay under the CBMF algorithm can be explicitly bounded as follows*

- *If the number of robots is no greater than the number of flows, i.e.* $0 < N \leq K$*,*

$$\bar{D} \leq (c_1 + 2)T \tag{9}$$

- *If the number of robots is greater than the number of flows , i.e.* $K < N \leq 2K$*,*

$$\bar{D} \leq \begin{cases} 2T & \text{if } \lambda \leq \frac{R_{avg}}{2} \\ \left\{\frac{R_{avg}}{2\lambda} + 1 + (c_2+2)(1-\frac{R_{avg}}{2\lambda})\right\}T & \text{if } \lambda > \frac{R_{avg}}{2} \end{cases} \tag{10}$$

where $c_1 = \lfloor \frac{K}{N} \rfloor$ *and* $c_2 = \lfloor \frac{K}{N-K} \rfloor$*, which are constants as* K *and* N *are given.*

Proof. Depending on the number of flows and the number of robots in the network, the delay analysis falls into one of the following two cases:

Case 1 when $0 < N \leq K$, the system follows Robot Allocation Strategy I in Algorithm 3 by Lemma 1. Since the arrival rate vector is within capacity region $\mathbf{\Lambda_{IB}}(\mathbf{v}, \mathbf{T})$, it will take a finite amount of time for the system to become stable. When it is stable, the total queue size in the system evolves exactly the same every two epochs. Assume the first epoch in a two-epoch cycle is the one when robots move to sources to collect data and the second of a two-epoch cycle is the one when robots move to sinks to deliver data. Denote the total queue size at the beginning of a two-epoch cycle in the system as Q_{cycle}, which remains the same every two epochs.

At the beginning of a new two-epoch cycle when N robots start to move to sources to collect data, the queue size at a most recent served source is λT. This is because a most recent served source is the one which is served in the last two-epoch cycle. And in the first epoch of the last two-epoch cycle, all data at the source can be collected by a robot, and in the second epoch of the last two-epoch cycle, as the robot delivers data to the corresponding sink, new data keep arriving and accumulating at the source to λT. The number of the most recent served sources is N, which is the same as the number of robots. Similarly, the data at a second most served source is $3\lambda T$, where the $2\lambda T$ additional data come from the fact that there is no robot serving it in the most recent two-epoch cycle. And the number of the second most recent served sources is also N. Thus, if we repeat this analysis, we can get the queue size of each robot. Denote $c_1 = \lfloor \frac{K}{N} \rfloor$, which is a constant as K and N are fixed. The number of sources with queue size $\lambda T, 3\lambda T, ..., (2c_1 - 1)\lambda T$ respectively is N and the number of sources with a queue size of $(2c_1 + 1)\lambda T$ is $K - c_1 N$. Since all robots have empty queues at the beginning of a two-epoch cycle, and sinks always have empty queues, the total queue at the start of a new two-epoch cycle in the system only consists of queues at source, which is

$$\begin{aligned} Q_{cycle} &= N\lambda T[1 + 3 + ... + (2c_1 - 1)] \\ &\quad + (K - c_1 N)(2c_1 + 1)\lambda T \\ &= [(2c_1 + 1)K - (c_1^2 + c_1)N]\lambda T \end{aligned} \tag{11}$$

In the first epoch of a two-epoch cycle when robots move to sources to collect data, since there is no data being delivered to sinks and each source has an arrival rate of λ, the total queue backlog in the system increases as $K\lambda t$. In the second epoch when robots move to sinks to deliver data, each source keeps having data arrive at rate λ. But due to the fact that data is being delivered to sinks from robots, the total queue growth rate is no greater than $K\lambda t$. Thus, assume the time needed for the system to become stable is n_1 epochs, then the total queue in the system at a time t after the system is stable satisfies

$$Q(t) \leq Q_{cycle} + K\lambda(t - n_1 T - 2T\lfloor \frac{t - n_1 T}{2T} \rfloor) \tag{12}$$

Thus the total accumulation of queues of all flows during a time interval $[0, (n_1 + 2n_2)T]$ that we are interested in becomes

$$\begin{aligned} &\int_0^{(n_1+2n_2)T} Q(t)dt \\ &= \int_0^{n_1 T} Q(t)dt + \int_{n_1 T}^{(n_1+2n_2)T} Q(t)dt \\ &\leq \int_0^{n_1 T} Q(t)dt + n_2 \int_0^{2T} (Q_{cycle} + K\lambda t)dt \end{aligned} \tag{13}$$

the last inequality comes from eqn. (12).

Therefore time average total queue in the system satisfies

$$\begin{aligned} \bar{Q} &= \lim_{n_2 \to \infty} \frac{1}{(n_1 + 2n_2)T} \int_0^{(n_1+2n_2)T} Q(t)dt \\ &\leq \lim_{n_2 \to \infty} \frac{1}{(n_1 + 2n_2)T} \left\{ \int_0^{n_1 T} Q(t)dt \right. \\ &\qquad \left. + n_2 \int_0^{2T} (Q_{cycle} + K\lambda t)dt \right\} \\ &\leq \frac{1}{2T} \int_0^{2T} (Q_{cycle} + K\lambda t)dt \end{aligned} \tag{14}$$

where the last inequality indicates that the time average total queue is upper bounded by the time average queue in a stable two-epoch cycle.

Taking eqn. (11) to eqn. (14) gives

$$\bar{Q} \leq [(2c_1 + 2)K - (c_1^2 + c_1)N]\lambda T \tag{15}$$

According to the Little's Theorem, the corresponding delay is $\bar{D} = \frac{\bar{Q}}{K\lambda}$, thus we have

$$\bar{D} \leq [2c_1 + 2 - \frac{(c_1^2 + c_1)}{\frac{K}{N}}]T \tag{16}$$

Further, since $c_1 = \lfloor \frac{K}{N} \rfloor < c_1 + 1$, we have

$$\bar{D} \leq (c_1 + 2)T \tag{17}$$

Case 2 when $K < N \leq 2K$, the system follows Robot Allocation Strategy II in Algorithm 4 by Lemma 2. According to Algorithm 4, robots are divided into two groups containing K and $N - K$ robots respectively. In addition, the serving patterns of these two group are complementary (or interchangeably). Specifically, if in an epoch when the group of K robots collect data from sources, the group of $N - K$ robots deliver data to sinks; Or if in an epoch when the group of K robots deliver data to sinks, the group of $N - K$ robots collect data from sources. Instead of randomly allocate a robot in Group I to a source to collect data in odd epochs in Algorithm 4, one feasible solution is to keep each robot in Group I serving the same source in all odd epochs[5]. Thus, depending on arrival rate, the corresponding delay of the network can be analyzed in two different subcases as follows

Subcase 1 when $0 \leq \lambda \leq \frac{R_{avg}}{2}$: only using the group of K robots can keep the network remain stable because they can take all data from sources in the collecting epoch,

5. The centralized coordinator can arbitrarily fix the robotic allocation of Group I in Algorithm 4 in such a way that keeps each robot in Group I serving the same source in all odd epochs. There might be other allocations of robots in Group I providing a better delay performance, but the theoretical result derived in Theorem 3 still holds.

and all the data can be delivered to sinks in the following delivering epoch. The other group of $N-K$ robots can help reduce delay in the network, but their different kinds of assignments result in the same network delay. Similarly, when the system is stable, the total queue size evolves exactly the same every two epochs. Let the first epoch of the two-epoch stable cycle be the one that the group of K robots finish delivering data to sinks and move to sources to collect data while the other group of $N-K$ robots finish collecting data from sources and move to sinks to deliver data; And let the second epoch of the two-epoch stable cycle be the one that the group of K robots finish collecting and move to sinks to deliver data while the other group of $N-K$ robots finish delivering and move to sources to collect data. Since every source has λT data arrive that have not been delivered to its sink, the total queue length in the system at the beginning of a two-epoch service cycle is $Q_{cycle} = K\lambda T$.

Since in a two-epoch cycle, there are always some robots delivering data, the total queue growth rate is less than $K\lambda$. Similar to the argument in Case 1 that the time average total queue in the system is upper bounded by the time average queue in a stable two-epoch cycle, therefore, we have

$$\begin{aligned}\bar{Q} &\leq \frac{1}{2T}\int_0^{2T}(Q_{cyce} + K\lambda t)dt \\ &\leq 2K\lambda T\end{aligned} \tag{18}$$

And

$$\bar{D} \leq 2T \tag{19}$$

Subcase 2 when $\lambda > \frac{R_{avg}}{2}$: after the group of K robots collecting data from sources, there can be leftovers in the sources which can drive the group of $N-K$ robots serve K flows in the same manner as Algorithm 3.

Therefore, the network can be considered as formed by two sub-networks. In one sub-network, there are K flows with arrival rate $\frac{R_{avg}}{2}$ and each flow has a designated robot serving it. In the second sub-network, there are K flows with arrival rate $\lambda - \frac{R_{avg}}{2}$ and $N-K$ robots providing service according to the Robot Allocation Strategy I in Algorithm 3. And serving patterns of these two sub-networks are complementary.

When the network is stable, the network evolves exactly the same every two epochs. Let the first epoch of the two-epoch stable cycle be the one that K robots move to sources to collect data in the first sub-network while $N-K$ robots move to sinks to deliver data in the second sub-network, and let the second epoch of the two-epoch stable cycle be the one that K robots move to sinks to deliver data in the first sub-network while $N-K$ robots move to sources to collect data in the second sub-network. Let $m = N-K$ and $\lambda_2 = \lambda - \frac{R_{avg}}{2}$. The queue of the first sub-network at the beginning of the two-epoch stable cycle is $Q_{cycle_1} = K\frac{R_{avg}}{2}T$ in the first sub-network.

For the second sub-network, following the similar analysis in Case 1, we can get the queue size of each robot at the beginning of one epoch before a new two-epoch cycle. Denote a constant as $c_2 = \lfloor \frac{K}{m} \rfloor$, then he number of sources with queue size $\lambda_2 T, 3\lambda_2 T, ..., (2c_2-1)\lambda_2 T$ respectively is m and the number of sources with a queue size of $(2c_2+1)\lambda_2 T$ is $K - c_2 m$ at the beginning of the prior epoch. Taking the new arrival data within the prior epoch into consideration, the total queue in the second sub-network at the start of a new two-epoch cycle is

$$\begin{aligned}Q_{cycle_2} &= m\lambda_2 T[1+3+...+(2c_2-1)] \\ &\quad + (K-c_2 m)(2c_2+1)\lambda_2 T + K\lambda_2 T \\ &= [(2c_2+2)K - (c_2^2+c_2)m]\lambda_2 T\end{aligned} \tag{20}$$

Thus, the total queue in the system at the beginning of a new two-epoch cycle is

$$\begin{aligned}Q_{cycle} &= Q_{cycle_1} + Q_{cycle_2} \\ &= K\frac{R_{avg}}{2}T + [(2c_2+2)K - (c_2^2+c_2)m]\lambda_2 T\end{aligned} \tag{21}$$

In a two-epoch cycle, there are always some robots delivering data, the total queue growth rate is less than $K\lambda$. Similar to the previous argument that the time average total queue in the system is upper bounded by the time average queue in a stable two-epoch cycle, then we have $\bar{Q} \leq \frac{1}{2T}\int_0^{2T}(Q_{cycle} + K\lambda t)dt$, which yields

$$\bar{Q} \leq K\frac{R_{avg}}{2}T + [(2c_2+2)K - c_2^2 m - c_2 m]\lambda_2 T + K\lambda T \tag{22}$$

And

$$\bar{D} \leq \{\frac{R_{avg}}{2\lambda} + [(2c_2+2) - \frac{(c_2^2+c_2)}{\frac{K}{m}}]\frac{\lambda_2}{\lambda} + 1\}T \tag{23}$$

Similarly, since $c_2 = \lfloor \frac{K}{m} \rfloor < c_2 + 1$ and $\lambda_2 = \lambda - \frac{R_{avg}}{2}$, we further have

$$\bar{D} \leq \left\{\frac{R_{avg}}{2\lambda} + 1 + (c_2+2)(1 - \frac{R_{avg}}{2\lambda})\right\}T \tag{24}$$

□

8 Simulations

We first present numerical simulation results for a network containing twenty flows and thirty robots. The sources and sinks of twenty flows are randomly located in a 200×200 2D plane, and all mobile robots initially start from the center of the plane, i.e., the location $(100, 100)$. Packets arrive at sources following the same deterministic process. In the simulation, we use a typical distance-rate function to represent the transmission rate. A reasonable first-order model can be obtained by combining Shannon's formula [31] for the capacity of an AWGN formula [31] as a function of its SNR. If the transmit power is kept constant, the SNR in turn depends upon how the signal power varies with distance. To account for near-field effects, we assume that this decay happens as $\frac{P_{max}}{1+d^\eta}$, where η is the path loss exponent, and P_{max} is the maximum received power. Thus the transmission rate as a function of distance is $R(d) = \log\left(1 + \frac{C}{1+d^\eta}\right)$, where C is a constant that takes into account the transmitter power as well as the noise variance at the receiver. We use $\eta = 2$, $C = 1$, v, T and λ are varied as shown in the figure 3.

In the figure 3 (left) we see the average end-to-end delay (it is obtained by measuring the average total queue size in the simulations and dividing by the total arrival rate, as per Little's Theorem [30]) versus arrival rate as we fix epoch duration T and vary velocity v, plotted wherever CBMF results in stable queues; we find that we are able to get converging, bounded delays (indicative of stability)

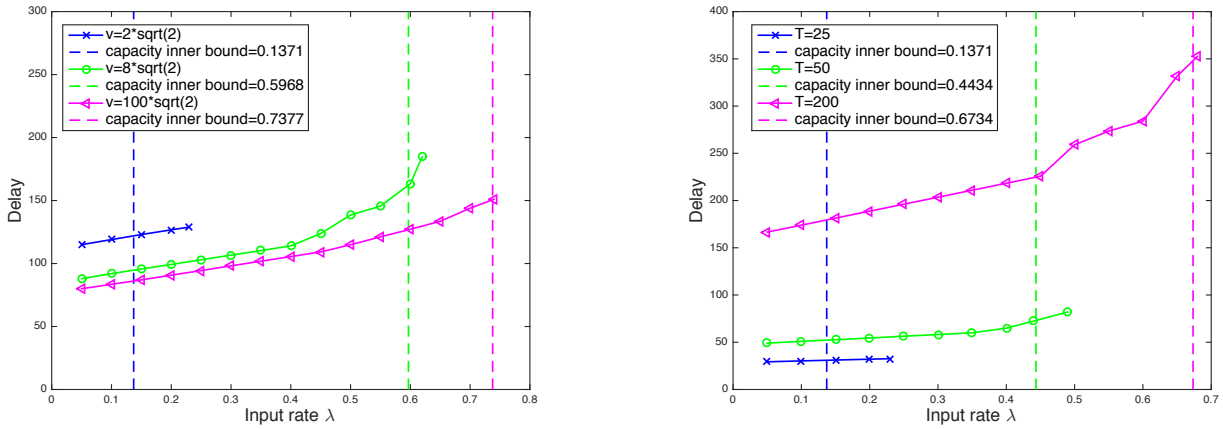

Fig. 3. Delay as we vary v for $T = 100$ (left) and delay as we vary T for $v = 8\sqrt{2}$ (right) for 20-Flows-30-Robots network

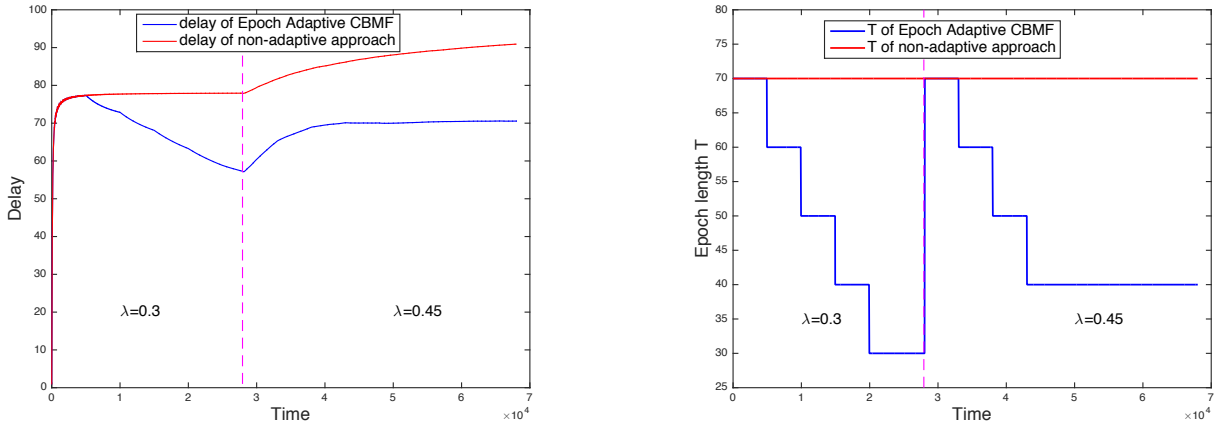

Fig. 4. Delay (left) and Epoch Duration (right) comparison of the Epoch Adaptive CBMF Algorithm with a non-adaptive scheme for 20-Flows-30-Robots network

even beyond the inner-bound capacity line. Also marked on the figure is the lower (inner) bound of capacity, for rates below which CBMF is provably stable. We see that as the velocity increases, so does the capacity, and at the same time the delay decreases. Thus improvement in robot velocity benefits both throughput and delay performance of CBMF, as may be expected. Figure 3 (right), in which the velocity v is kept constant across curves but the epoch duration T is varied, is somewhat similar but with one striking difference, however, as the epoch duration increases, so does the capacity; but at the same time, the average delay also increases (for the same arrival rates, so long as stability is maintained). Thus, increasing the scheduling epoch duration improves throughput but hurts delay performance.

Next, to show the performance of the Epoch Adaptive CBMF in Algorithm 2, we conduct a simulation of a network whose settings are the same as the previous 20-Flows-30-Robots network. We use $T_{th} = 70$, $L = 5000$ and $p = 0.1$. Two different algorithms have been implemented to set the epoch length T: one is to apply the Epoch Adaptive CBMF in Algorithm 2, and the other is a non-adaptive scheme where T is fixed as T_{th} all the time. We present the delay performance (in Figure 4 left) of these two approaches together with the corresponding epoch length at each time (in Figure 4 right). Initially the arrival rate is set as $\lambda = 0.3$. As is shown in Figure 4, the Epoch Adaptive CBMF reduces the epoch duration to improve network delay while keeping the network stable. When there is an increase in the arrival rate where λ becomes 0.45, the Epoch Adaptive CBMF algorithm can detect this change and adapt T accordingly to keep the network stable while maintaining a small delay. Overall, applying our heuristic algorithm to adjust the epoch length according to current network condition can provide a better delay performance.

Finally, we evaluate the performance of the CBMF algorithm in a multi-flow homogeneous network. Particularly, we are interested in how delay changes with respect to the epoch duration T, which is tunable to meet different network conditions when applying CBMF algorithm in practice. We consider two different network settings: a 20-Flows-10-Robots network and a 20-Flows-30-Robots network. As can be seen in Figure 5, the delay performance in simulation is bounded by our theoretical analysis. We also present the results of how delay changes according to epoch length T in Figure 6. As it is shown, the delay grows linearly with the epoch length T in both simulation and theory. In

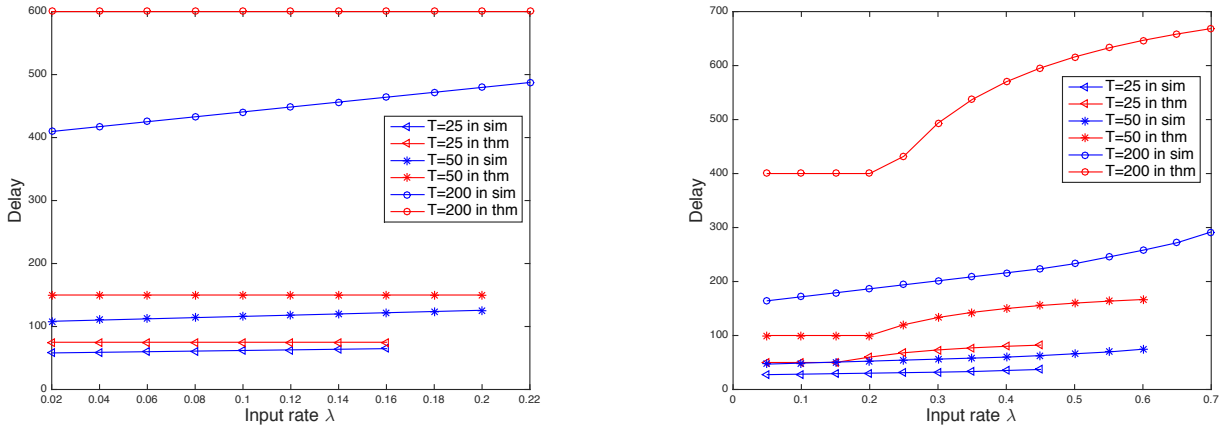

Fig. 5. Delay of as we vary λ for homogeneous 20-Flows-10-Robots network (left) and 20-Flows-30-Robots network (right)

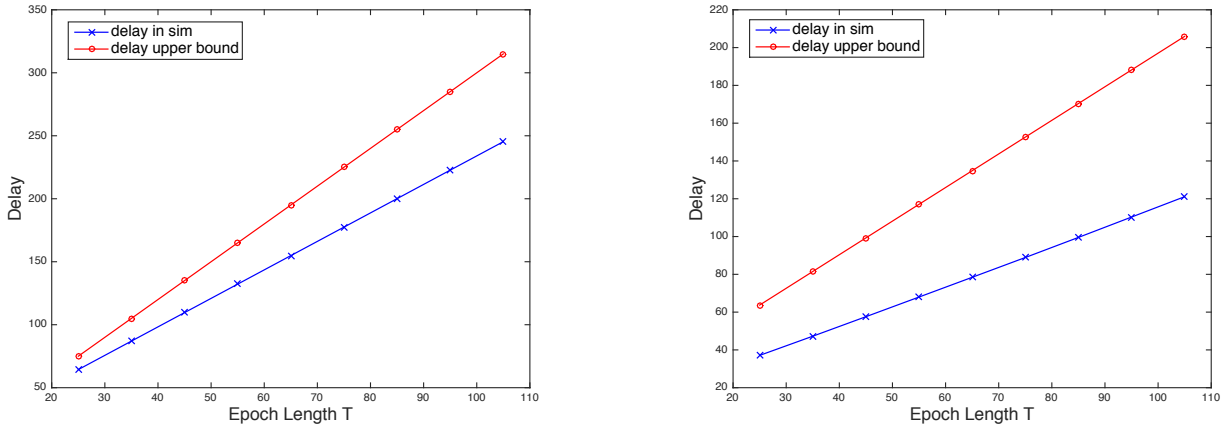

Fig. 6. Delay of as we vary T for a homogeneous 20-Flows-10-Robots network (left) and 20-Flows-30-Robots network (right)

addition, during the running of our simulation, we make an observation of the allocation of robots at each epoch and confirm that the CBMF algorithm indeed matches the robot allocation strategy in Algorithm 3 or 4, which indicates the CBMF has a simple time sharing structure.

9 Conclusions and Future Work

This paper has addressed three fundamental questions in robotic message ferrying for wireless networks: what is the throughput capacity region of such systems? How can they be scheduled to ensure stable operation, even without prior knowledge of arrival rates? And what is the corresponding delay performance, and is there a way to maintain stability while having small delay in realistic network settings?

We have mathematically characterized the capacity region of such systems in both ideal and realistic settings. A dynamic CBMF algorithm has been proposed to schedule robots to guarantee the network stability even without knowing the arrival rates. We have derived an inner bound of the capacity region together with its corresponding delay performance that the CBMF algorithm can achieve. The fact that the delay scales linearly with the epoch duration has guided us to design a heuristic approach to adapt epoch duration during run time to improve the delay performance while keeping network stable. It has also been shown that in a multi-flow homogeneous network, the CBMF algorithm with two additional preferences has a simple time sharing structure, which is easy to implement in practice.

There are a number of open directions suggested by the present work. The first is to improve the CBFM algorithm to support the entire capacity region without delay inefficiency, possibly by considering a finer-grained motion control as a generalization of [29] to improve the utilization of robots. To be specific, currently we assume each robot can only be assigned to one node for the whole epoch. Even if some robots may finish their tasks earlier than the others, they can not be re-assigned to serve other nodes in this epoch. A generalization of [29] can take care of this scenario by using a much finer-grained motion control such that the allocation decision is made at each time slot. This algorithm is conceptually similar to the traditional Backpressure routing algorithm. However, the capacity analysis in the traditional backpressure formulation is based on the assumption that the link states / network topology is independent of the nodes' allocation, whereas in our problem, the robots' allocations can change the link states (i.e., the communication rates). This coupling between allocation and

link rate makes the problem more complicated and does not allow us to follow the standard analysis using backpressure formulation. Therefore, the problem becomes non-trivial, and in our prior work [29] on this topic, we were able to solve the problem only for a single flow, single robot case. In general, there is no certain answer.

Second, we would like to design more applicable algorithms that take additional practical factors into consideration. In addition to finite buffer size and packet loss that have been discussed before, communication interference and movement collision should also be taken care of in practice. In this work, we assume that communication pairs use orthogonal channels to avoid interference, the problem becomes more interesting and practical when interference is taken into consideration. In that case, allocations with interfering communication pairs are not allowed. Careful path planning for mobile robots is also required to avoid two robots moving closes that causes interference. Movement collision between robots might happen and affect the system performance, especially in the case when there are a large number of mobile robots in the network. Additional collision avoidance method needs to be taken into consideration to avoid collision. All these additional considerations make the problem non-trivial and the same current framework will not be applicable.

Finally, though our current centralized framework is relatively inexpensive since only queue state information needs to be collected and the CBMF has a polynomial runtime, it is still not robust and realistic in practice, thus we are interested in developing decentralized scheduling mechanisms that the robots can implement in a distributed fashion. Also, we plan to find more structural properties of the CBMF algorithm and its delay performance in general cases.

Acknowledgment

This work was funded in part by the National Science Foundation, under award number 1423624.

References

[1] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Ad Hoc Wireless Networks," *IEEE/ACM Trans. on Networking*, vol 10, no 4, August 2002.

[2] W. Zhao and M.H. Ammar, "Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks," *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, May 2003.

[3] D. Jea, A. Somasundara, and M. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in sensor networks," *IEEE DCOSS*, 2005.

[4] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," *IEEE INFOCOM*, 2005.

[5] D. K. Goldenberg, J. Lin, S. A. Morse, B. E. Rosen, and R. Y. Yang, "Towards mobility as a network control primitive," *ACM MobiHoc*, 2004.

[6] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Autom. Control*, vol.51, no.3, pp.401420, 2006.

[7] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol.49, no.9, pp.14651476, 2004.

[8] V.GaziandK.Passino, "Stability analysis of swarms," *IEEE Trans. Autom. Control*, vol.48, no.4, pp.692697, 2003.

[9] Y. Yan and Y. Mostofi, "Robotic router formation in realistic communi-cation environments," *IEEE Trans. Robotics*, vol.28, no.4, pp.810827, 2012.

[10] M. Zavlanos, A. Ribeiro and G. Pappas,"Mobility and routing control in networks of robots," *CDC*, 2010.

[11] R. Williams, A. Gasparri and B. Krishnamachari, "Route Swarm: Wireless Network Optimization through Mobility," *IROS*, 2014.

[12] L. Tassiulas, A. Ephremides, "Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, Vol. 37, No. 12, pp. 1936-1949, 1992.

[13] H. W. Kuhn and B. Yaw, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955

[14] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, 2006.

[15] M. Neely and R. Urgaonkar, "Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage," *IEEE SSC*, 2008.

[16] M. Neely, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM TON*, 2008.

[17] M. Neely, "Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks," *IEEE TAC*, 2009.

[18] A. Dvir and A. V. Vasilakos, "Backpressure-based routing protocol for DTNs," *ACM SIGCOMM*, 2010.

[19] M. Neely, E. Modiano and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE Transactions on Networking*, volv. 16, No. 2, pp. 396-409, 2008.

[20] M. J. Neely, "Energy-Aware Wireless Scheduling with Near Optimal Backlog and Convergence Time Tradeoffs,", *IEEE INFOCOM*, 2015.

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

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

[23] L. Huang, S. Moeller, M. Neely, and B. Krishnamachari, "LIFO- backpressure achieves near-optimal utility-delay tradeoff," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 831844, June 2013.

[24] S. Supittayapornpong and M. J. Neely, "Achieving Utility-Delay-Reliability Tradeoff in Stochastic Network Optimization with Finite Buffers," *IEEE INFOCOM*, 2015

[25] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "Diffq: Practical differential backlog congestion control for wireless networks," *IEEE INFOCOM*, 2009.

[26] A. Sridharan, S. Moeller, and B. Krishnamachari, "Implementing backpressure-based rate control in wireless networks," *ITA Workshop*, 2009.

[27] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: the backpressure collection protocol," *ACM/IEEE IPSN*, 2010.

[28] S. Wang, A. Gasparri, B. Krishnamachari, "Robotic Message Ferrying for Wireless Networks using Coarse-Grained Backpressure Control," short paper in IEEE Globecom 2013 Workshop - Wireless Networking and Control for Unmanned Autonomous Vehicles.

[29] A. Gasparri and B. Krishnamachari, "Throughput-optimal robotic message ferrying for wireless networks using backpressure control," in *IEEE MASS*, 2014.

[30] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison-Wesley, 1993.

[31] A. Goldsmith, *Wireless Communications*, Cambridge Univ. Press, 2005.

Shangxing Wang received the B.S. degree in electrical engineering at Xidian University, Xian, Shaanxi, China, in 2010. She is currently pursuing the Ph.D. degree in the Autonomous Networks Research Group at the University of Southern California, Los Angeles, CA, USA.

Her research interest is in the area of robotic wireless networks and stochastic network optimization.

Andrea Gasparri Andrea Gasparri (M09) received the cum laude degree in computer science and the Ph.D. degree in computer science and automation, both from the University of Roma Tre, Rome, Italy, in 2004 and 2008, respectively.

He has been a Visiting Researcher at several institutions, including the Universite Libre de Bruxelles, Brussel, Belgium, City College of New York, New York, NY, USA, and the University of Southern California, Los Angeles, CA, USA. He is currently an Assistant Professor with the Department of Engineering, University of Roma Tre. His current research interests include mobile robotics, sensor networks, and networked multi-agent systems.

Dr. Gasparri was the recipient of the Italian grant FIRB Futuro in Ricerca 2008 for the project Networked Collaborative Team of Autonomous Robots funded by the Italian Ministry of Research and Education (MIUR).

Bhaskar Krishnamachari Bhaskar Krishnamachari received his B.E. in Electrical Engineering at The Cooper Union, New York, in 1998, and his M.S. and Ph.D. degrees from Cornell University in 1999 and 2002 respectively.

He is a Professor in the Department of Electrical Engineering at the University of Southern California's Viterbi School of Engineering. His primary research interest is in the design and analysis of algorithms and protocols for next-generation wireless networks.

Supplemental Material: Robotic Message Ferrying for Wireless Networks using Coarse-Grained Backpressure Control

Shangxing Wang, Andrea Gasparri, *Member, IEEE,* and Bhaskar Krishnamachari, *Member, IEEE*

APPENDIX A
PROOF OF THEOREM 1

To prove this theorem, we need to show that the arrival rate region $\mathbf{\Lambda}$ can be served by a convex combination of basis configurations in which robots are allocated to serve distinct flows.

Let us refer to the convex hull of $\tilde{\Gamma}$ as $\mathcal{H}(\tilde{\Gamma})$ or, for readability, simply $\mathcal{H}$. Then we have the follwing lemma:

Lemma 1. $\mathcal{H} \supset \mathbf{\Lambda}$

Proof. First, note that the convex hull of $\tilde{\Gamma}$ can be written as follows:

$$\mathcal{H} = \left\{ \gamma | \gamma_i = \frac{a_i R_{max}}{2}, a_i \in [0,2] \quad \forall\, i, \sum_{i=1}^{K} a_i \leq N \right\} \quad (1)$$

In other words, the convex hull of the set $\tilde{\Gamma}$ is obtained by allowing a_i to vary continuously. Now using the relationship $a_i = \frac{2\gamma_i}{R_{max}}$, we can re-express $\mathcal{H}$ as follows:

$$\begin{aligned} \mathcal{H} &= \left\{ \gamma | \frac{2\gamma_i}{R_{max}} \in [0,2] \; \forall\, i, \sum_{i=1}^{K} \frac{2\gamma_i}{R_{max}} \leq N \right\} \\ &= \left\{ \gamma | 0 \leq \gamma_i \leq R_{max} \; \forall\, i, \sum_{i=1}^{K} \gamma_i \leq \frac{R_{max} N}{2} \right\} \supset \mathbf{\Lambda} \end{aligned}$$

□

Each basis allocation corresponding to the elements of $\tilde{\Gamma}$ can actually be expressed as two distinct but symmetric allocations of robots to sources/sinks over two successive epochs. For the i^{th} flow, if $a_i = 0$, there is no robot allocated to either the source or sink in either of these two epochs; if $a_i = 1$, a particular robot is assigned to be at the source at the first epoch and at the sink at the second epoch; if $a_i = 2$, two robots are assigned (call them R_1 and R_2) such that R_1 is at the source at the first epoch and at the sink at the second epoch while R_2 is at the sink at the first epoch and at the source at the second epoch.

The set $\mathcal{H}$ describes all possible robot service rates that can be obtained by a convex combination of these basis allocations. Consider a rate vector $\gamma \in \mathcal{H}$. Since it lies in the convex hull of the set $\tilde{\Gamma}$ it can be described in terms of a vector of convex coefficients α each of whose elements corresponds to a basis allocation of robots. We can therefore[1] identify n_i such that $n_i / \sum_i n_i = \alpha_i$. The given rate vector γ can then be scheduled by allocating n_i epochs each for the two parts of the i^{th} basis allocation. And after a total of $\sum_i 2n_i$ epochs, the whole schedule can be repeated. This schedule will provide the desired service rate vector γ.

Thus far the schedules have been derived under the assumption of instantaneous robot movements. Now we consider the effect of transit time. It is possible to choose T or v to be sufficiently large to bound the fraction of time spent in transit by ϵ, i.e. $\frac{d_{max}}{vT} < \epsilon$, where d_{max} is the maximum distance between static nodes. Thus even while taking into account time wasted in transit, we can scale either time period of the epochs T or the velocity v so as to provide a service rate vector γ' that is arbitrarily close to any ideal service rate γ in the sense that $\gamma_i - \gamma_i' < \epsilon \; \forall\, i$.

Together, these imply that $\mathbf{\Lambda}$ is the achievable capacity region of the network.

APPENDIX B
PROOF OF THEOREM 2

The main idea to prove this theorem is to show the time average total queue in the system can be upper bounded.

From an average point of view, alternatively, we can consider this mobile network as a static network where robots are static and have a constant transmission rate as R_{avg}. As R_{avg} is the worst-case average transmission rate, some robots may have higher average transmission rates if their travelling distances are smaller than d_{max} in some epochs. In that case, we can assume those robots pause communicating with their assigned nodes for some time during the epochs, and thus their average transmission rates

- *S. Wang and B. Krishnamachari are with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, 90007.*
  E-mail: shangxiw@usc.edu and bkrishna@usc.edu
- *A. Gasparri is with the Department of Engineering, Roma Tre University, Rome, Italy.*
  E-mail: gasparri@dia.uniroma3.it

1. Here, for ease of exposition, we are assuming that α_i is rational, otherwise it can be approximated by an arbitrarily close rational number which will not affect the overall result.

can still be the same as R_{avg}. This assumption also indicates why the capacity region $\mathbf{\Lambda_{IB}}(\mathbf{v},\mathbf{T})$ we are going to prove in the following is only an inner bound since some robots are under utilization under the assumption. And the CBMF algorithm can actually achieve a better capacity region in practice.

Let $b_{ij}(t) \in \{0,1\}$ represent if a robot j is allocated to source $src(i)$. $b_{ij}(t) = 1$ indicates robot j is allocated to $src(i)$ and $b_{ij}(t) = 0$ indicates robot j is not allocated to $src(i)$. Similarly, $c_{ij}(t) \in \{0,1\}$ represents whether a robot j is allocated to sink $sink(i)$. Since at any time t a robot can be allocated to exactly one source or sink, we have $\sum_{i=1}^{K} b_{ij}(t) + c_{ij}(t) = 1$. The transmission rates from a $src(i)$ to robot j and from robot j to $sink(i)$ are $R_{src(i),j}(t) = b_{ij}(t)R_{avg}$ and $R_{j,sink(i)}(t) = c_{ij}(t)R_{avg}$ respectively.

At the beginning of epoch $n+1$ (before making a new allocation), the queue backlog at source i, $\forall i \in \{1,...,K\}$, is updated as follows

$$\begin{aligned} Q_{src(i)}((n+1)T) = max\{&Q_{src(i)}(nT) + \lambda_i(T-1) \\ &- \sum_{j=1}^{N} b_{ij}(nT)R_{avg}T, 0\} + \lambda_i \end{aligned} \tag{2}$$

The queue backlog at robot j for flow i at the beginning of epoch $(n+1)T$, $\forall i \in 1,...,K$ and $j \in 1,...,N$, is given by

$$\begin{aligned} Q_j^i((n+1)T) = &max\{Q_j^i(nT) - c_{ij}(nT)R_{avg}T, 0\} \\ &+ min\{Q_{src(i)}(nT) + \lambda_i(T-1), \\ &\quad b_{ij}(nT)R_{avg}T\} \end{aligned} \tag{3}$$

Define the queue backlog vector of this system at the beginning of epoch n as

$$\begin{aligned} \mathbf{\Theta}(nT) = \big(&Q_{src(1)}(nT),...,Q_{src(K)}(nT), Q_1^1(nT),..., \\ &Q_1^K(nT),...,Q_N^1(nT),...,Q_N^K(nT)\big) \end{aligned} \tag{4}$$

And the Lyapunov function as

$$L(\mathbf{\Theta}(nT)) = \frac{1}{2}\left[\sum_{i=1}^{K} Q_{src(i)}(nT)^2 + \sum_{i=1}^{K}\sum_{j=1}^{N} Q_j^i(nT)^2\right] \tag{5}$$

Then we have,

$$\begin{aligned} &L(\mathbf{\Theta}((n+1)T)) - L(\mathbf{\Theta}(nT)) \\ &\le \sum_{i=1}^{K} \frac{\left[\sum_{j=1}^{N} b_{ij}(nT)R_{avg}T - \lambda_i(T-1)\right]^2 + \lambda_i^2}{2} \\ &\quad + \sum_{i=1}^{K}\sum_{j=1}^{N} \frac{\left(c_{ij}(nT)R_{avg}T\right)^2 + \left(b_{ij}(nT)R_{avg}T\right)^2}{2} \\ &\quad + \sum_{i=1}^{K} Q_{src(i)}(nT)\left[\lambda_i - \sum_{j=1}^{N} b_{ij}(nT)R_{avg})\right]T \\ &\quad + \sum_{i=1}^{K}\sum_{j=1}^{N} Q_j^i(nT)\left(b_{ij}(nT) - c_{ij}(nT)\right)R_{avg}T \end{aligned} \tag{6}$$

where the inequality comes from equations (2) and (3), and

$$\left(max\left\{Q-b,0\right\}+a\right)^2 \le Q^2 + a^2 + b^2 + 2Q(a-b). \tag{7}$$

$$\left(max\left\{Q_1-c,0\right\}+min\left\{Q_2,b\right\}\right)^2 \le max\left\{Q_1-c,0\right\}+b \tag{8}$$

Define the conditional Lyapunov drift as

$$\triangle\left(\mathbf{\Theta}(nT)\right) = \mathbb{E}\left\{L(\mathbf{\Theta}((n+1)T)) - L(\mathbf{\Theta}(nT))|\mathbf{\Theta}(nT)\right\} \tag{9}$$

Based on the assumption that at any time at most one robot can be allocated to serve a source i ($\forall i \in \{1,...,K\}$), we have among all binary variables b_{ij} ($\forall j \in \{1,...,N\}$), at most one variable can be 1 and all the others are 0s. Thus, $\left[\left(\sum_{j=1}^{N} b_{ij}(nT)R_{avg}T\right)\right]^2 \le (R_{avg}T)^2$ and $\sum_{j=1}^{N}(b_{ij}(nT)R_{avg}T)^2 \le (R_{avg}T)^2$. Similarly, since at any time at most one robot can be allocated to serve a sink i ($\forall i \in \{1,...,K\}$), among all binary variables c_{ij} ($\forall j \in \{1,...,N\}$), at most one variable can be 1 and all the others are 0s. And we have $\sum_{j=1}^{N}(c_{ij}(nT)R_{avg}T)^2 \le (R_{avg}T)^2$. Since N, T and $R_{avg} \le R_{max}$ are all finite, the first and second moments of the data arrival process are finite, then we can define a finite constant B as

$$\begin{aligned} B = &\sum_{i=1}^{K} \frac{(R_{max}T)^2 + \lambda_i^2}{2} \\ &+ \sum_{i=1}^{K} \frac{(R_{max}T)^2 + (R_{max}T)^2}{2} \end{aligned} \tag{10}$$

which provides an upper bound for the first two terms on the right hand side (RHS) of inequality (6).

Thus we have,

$$\begin{aligned} &\triangle\left(\mathbf{\Theta}(nT)\right) \\ &\le B + \sum_{i=1}^{K} Q_{src(i)}(nT)\lambda_i T \\ &\quad - \sum_{i=1}^{K}\sum_{j=1}^{N} \mathbb{E}\left\{\left[(Q_{src(i)}(nT) - Q_j^i(nT))b_{ij}(nT)\right.\right. \\ &\qquad \left.\left. + Q_j^i(nT)c_{ij}(nT)\right]R_{avg}T|\mathbf{\Theta}(nT)\right\} \end{aligned} \tag{11}$$

Applying the CBMF algorithm to allocate robots, the last term on the RHS of (11) can be maximized, thus the conditional drift can be minimized. Let $b_{ij}^*(t)$ and $c_{ij}^*(t)$ represent any other robot allocation, then equation (11) can be re-written as

$$\begin{aligned} &\triangle\left(\mathbf{\Theta}(nT)\right) \\ &\le B - \sum_{i=1}^{K} Q_{src(i)}(nT)\left(\mathbb{E}\left\{\sum_{j=1}^{N} b_{ij}^*(nT)R_{avg}|\mathbf{\Theta}(nT)\right\} - \lambda_i\right)T \\ &\quad - \sum_{i=1}^{K}\sum_{j=1}^{N} Q_j^i(nT)\mathbb{E}\left\{\left(c_{ij}^*(nT) - b_{ij}^*(nT)\right)R_{avg}T|\mathbf{\Theta}(nT)\right\} \end{aligned} \tag{12}$$

In order to upper bound (12), let us first consider the following problem: given an arrival rate vector $\boldsymbol{\lambda} = (\lambda_1,...,\lambda_K) \in \mathbf{\Lambda_{IB}}(\mathbf{v},\mathbf{T})$, we want to design an S-only (depends only on the channel states) algorithm such that

$$\begin{aligned} \text{find} \quad & \epsilon > 0 \\ \text{subject to} \quad & \lambda_i + \epsilon \le \mathbb{E}\left\{\sum_{j=1}^{N} b_{ij}^*(t)R_{avg}\right\},\ \forall i \qquad (a) \\ & \mathbb{E}\left\{b_{ij}^*(t)R_{avg}\right\} + \epsilon \le \mathbb{E}\left\{c_{ij}^*(t)R_{avg}\right\},\ \forall i,j \qquad (b) \end{aligned} \tag{P2}$$

Similar to the previous robots' allocation policy when prior knowledge about arrival rate vector is given in Section III, define the set of all possible robot service rates as $\mathcal{H}' = \left\{\gamma | \gamma_i = \frac{a_i R_{avg}}{2}, a_i \in [0,2] \quad \forall\, i, \sum_{i=1}^{K} a_i \leq N\right\}$. Then an S-only algorithm to achieve any given arrival rate vector strictly interior to $\mathcal{H}'$ can be designed as follows:

Since $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_K) \in \mathcal{H}'/\partial\mathcal{H}'$, we can find a vector $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_K)$ such that $\boldsymbol{\lambda}' = (\lambda_1 + \epsilon_1, ..., \lambda_K + \epsilon_K) \in \partial\mathcal{H}'$. Let $\epsilon_{max}(\boldsymbol{\lambda}) = \min\{\epsilon_1, ..., \epsilon_K\}$, and since λ is strictly interior in $\mathcal{H}'$, we have $\epsilon_{max}(\boldsymbol{\lambda}) > 0$.

Let $\boldsymbol{\lambda}'' = (\lambda_1 + \epsilon_{max}(\boldsymbol{\lambda}), ..., \lambda_K + \epsilon_{max}(\boldsymbol{\lambda})) \in \mathcal{H}'$, and it can be represented as a convex combination of basis allocations in $\mathcal{H}'$. To be specific, in a network containing K flows and N robots, there are M (depending on K and N) basis allocations in total. Let $(\lambda_{l1}, ..., \lambda_{lK})$, $\forall l \in \{1, ..., M\}$ denote the capacity the l_{th} allocation can provide. Let $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_M)$ be the allocation vector of the convex coefficients such that $\forall l \in 1, \ldots, M$, $\alpha_i \geq 0$ and $\sum_{l=1}^{M} \alpha_l = 1$. Then we have

$$\begin{aligned} &\alpha_1(\lambda_{11}, ..., \lambda_{1K}) + ... + \alpha_M(\lambda_{M1}, ..., \lambda_{MK}) \\ &= (\lambda_1 + \epsilon_{max}(\boldsymbol{\lambda}), ..., \lambda_K + \epsilon_{max}(\boldsymbol{\lambda})) \end{aligned} \tag{13}$$

After finding integers n_l such that $n_l / \sum_{j=1}^{M} n_j = \alpha_l$, $\forall l \in \{1, ..., M\}$, the arrival rate vector $\boldsymbol{\lambda}''$ can be served by first allocating n_l epochs for the l_{th} basis allocation, and allocating the next n_l epochs for the same l_{th} basis allocation but exchanging the robots locations, $\forall l \in \{1, ..., M\}$. And after every $2\sum_{l=1}^{M} n_l$ epochs, repeat the whole process.

The above algorithm can guarantee to find a $\epsilon_{max}(\boldsymbol{\lambda}) > 0$ in (P2) with constraint (a) satisfied. But constraint (b) in P(2) cannot be met since the above algorithm makes $\mathbb{E}\left\{b_{ij}^*(t) R_{avg}\right\} = \mathbb{E}\left\{c_{ij}^*(t) R_{avg}\right\}, \forall i, j$. To satisfy constraint (b) in (P2), we can change the above algorithm by adding a few more epochs to each $2\sum_l n_l$ epochs period, during which we only have robots at sinks to help deliver data. In this way we can find a $\epsilon'(\boldsymbol{\lambda}) > 0$ that solves (P2), and this allows to express equation (12) as

$$\triangle(\boldsymbol{\Theta}(nT)) \leq B - \epsilon'(\boldsymbol{\lambda}) T \left[\sum_{i=1}^{K} Q_{src(i)}(nT) + \sum_{i=1}^{K}\sum_{j=1}^{N} Q_j^i(nT)\right] \tag{14}$$

Taking iterated expectations, summing the telescoping series, and rearranging terms yields:

$$\begin{aligned} &\sum_{k=0}^{n-1} \left[\sum_{i=1}^{K} \mathbb{E}\left\{Q_{src(i)}(kT)\right\} + \sum_{i=1}^{K}\sum_{j=1}^{N} \mathbb{E}\left\{Q_j^i(kT)\right\}\right] \\ &\leq \frac{nB}{\epsilon'(\boldsymbol{\lambda})T} + \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))\}}{\epsilon'(\boldsymbol{\lambda})T} \end{aligned} \tag{15}$$

Consider a time slot t in some epoch interval $[kT, (k+1)T]$, for every flow i with arrival rate λ_i, the total queue length of its packets satisfies

$$Q_{src(i)}(t) + \sum_{j=1}^{N} Q_j^i(t) \leq Q_{src(i)}(kT) + \sum_{j=1}^{N} Q_j^i(kT) + \lambda_i(t - kT) \tag{16}$$

Thus, the total accumulation of queues of all flows during time interval $[0, nT-1]$ satisfies

$$\begin{aligned} &\sum_{\tau=0}^{nT-1} \left[\sum_{i=1}^{K} \mathbb{E}\left\{Q_{src(i)}(\tau)\right\} + \sum_{i=1}^{K}\sum_{j=1}^{N} \mathbb{E}\left\{Q_j^i(\tau)\right\}\right] \\ &\leq \frac{nB(T-1)}{\epsilon'(\boldsymbol{\lambda})T} + \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))(T-1)\}}{\epsilon'(\boldsymbol{\lambda})T} + \sum_{i=1}^{K} \frac{T(T-1)n\lambda_i}{2} \end{aligned} \tag{17}$$

where the inequality comes from eqn. (15).

Therefore time average total queue in the system satisfies

$$\begin{aligned} \bar{Q} &= \lim_{n\to\infty} \frac{1}{nT} \sum_{\tau=0}^{nT-1} \left[\sum_{i=1}^{K} \mathbb{E}\left\{Q_{src(i)}(\tau)\right\} + \sum_{i=1}^{K}\sum_{j=1}^{N} \mathbb{E}\left\{Q_j^i(\tau)\right\}\right] \\ &\leq \frac{B(T-1)}{\epsilon'(\boldsymbol{\lambda})T^2} + \sum_{i=1}^{K} \frac{(T-1)\lambda_i}{2} \end{aligned} \tag{18}$$

which indicates the time average total queue is bounded and the system is strongly stable as B, T, $\boldsymbol{\lambda}$ and $\epsilon'(\boldsymbol{\lambda})$ are positive constants and K and N are fixed.

Further according to eqn. (10), $B = O(T^2)$. Then for any given $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}_{\mathbf{IB}}(\mathbf{v}, \mathbf{T})$, the time average total queue satisfies $\bar{Q} = O(T)$ so long as the system is stable. As per Little's Theorem [29]), the end-to-end delay is obtained by dividing the average total queue size by the total arrival rate, which gives $\bar{D} = \frac{\bar{Q}}{\sum_{i=1}^{K} \lambda_i} = O(T)$.

APPENDIX C
PROOF OF LEMMA 1

What we are interested in showing in the following is that under the homogeneous condition, the Robot Allocation Strategy I indeed follows the CBMF algorithm that solves the optimization problem (P1) at the start of each epoch.

In the initial epoch, since all queues are empty, the centralized scheduler prefers assigning each robot to a source to collect data and all data from each served source can be collected at the end of this epoch. Thus at the end of the initial epoch, each robot contains λT data in its queue corresponding to its assigned source. At the end of the first epoch, there are N sources served whose queues are empty and each of the $K - N$ unserved sources has a queue size of λT. Thus, at the beginning of the second epoch, though the optimization problem (P1) may have multiple solutions that all give the maximum total queue differential $N\lambda T$, based on our assumptions, the scheduler prefers choosing the solution that assign all robots to sinks with the purpose to reduce delay.

In the second epoch, robots can deliver all their data to sinks and their queue become empty at the end. Thus, the queue differential between any robot and any sink is 0. Additionally, each source has λT new data arrive during this epoch. Thus, at the beginning of the third epoch, according to the CBMF algorithm, all robots need to be allocated to sources and the corresponding allocation A_3[2] solves the optimization problem (P1) with maximized total queue differential as $\sum_{i,j} w(A_3(i,j))$. During the third epoch, robots can collect all data from their assigned sources. At the

2. A_i represents the allocation in epoch i.

beginning of the forth epoch, if one assign all robots to sinks associated with the previously served sources, the total queue differential is $\sum_{i,j} w(A_3(i,j)) + N\lambda T$, which is the maximum total queue differential. This is because at the beginning of the third epoch, the total queue differential $\sum_{i,j} w(A_3'(i,j))$ provided by any other allocation A_3' is no greater than $\sum_{i,j} w(A_3(i,j))$. During the third epoch, each source has an amount of λT new data arrive, and at most N sources can have their new data arrived in the third epoch completely being collected by their serving robots. Therefore, at the beginning of the fourth epoch, any allocation cannot provide a total queue differential greater than $\sum_{i,j} w(A_3(i,j)) + N\lambda T$. Thus, in the fourth epoch, based on the CBMF, the allocation that assigns robots to sinks associated with the sources they serve in the third epoch is the solution to the optimization problem (P1).

If we apply the above analysis for all epochs, it can be proved that robots work as one single serving group and serve sources and sinks in alternative cycles under the CBMF algorithm. In addition, since all flows are homogeneous, each flow has an equivalent amount of time being served. This further indicates that in every odd epoch, robots need to be allocated to least recent served sources to collect data, and in the following even epoch, robots need to be assigned to the corresponding sinks to deliver data. Thus, the robotic allocation turns out to follow the Robot Allocation Strategy I shown in Algorithm 3.

APPENDIX D
PROOF OF LEMMA 2

Similar to the proof of Lemma 1, what we are going to show is that the Robot Allocation Strategy II stated in Lemma 2 follows the CBMF algorithm that solves the optimization problem (P1) at the start of each epoch.

In the first epoch, since all queues in the system are empty, the allocation is to allocate K robots to move to sources to collect data and the other $N-K$ robots to move sinks to deliver data (though the robots do not have any data during this initial epoch). The arrival rate of each source is λ. At the end of the first epoch, each of the K collecting robot contains λT data in its queue associated with its assigned source, and all other nodes or robots have queue size 0. Thus, at the beginning of the second epoch, the allocation that solves the optimization problem (P1) is to allocate each previous collecting robot to move to the corresponding sink to deliver data and the other $N-K$ previous delivering robots to sources to collect data.

In the second epoch, each of the $N-K$ robots can collect all data from its assigned source and each of the K delivering robots can deliver all its collected data to its sinks. Thus, at the end of the second epoch, each of the $N-K$ robots has a queue size λT associated with its assigned source, and the queue of each of the K delivering robot becomes empty. Among K sources, there are $N-K$ sources having been served whose queues are empty and each of the $2K-N$ unserved sources has a queue size of λT. Thus, at the beginning of the third epoch, based on the CBMF, the allocation A_3 that solves the optimization problem (P1) with sinks preferred is to allocate the $N-K$ robots to corresponding sinks to deliver data and the K robots to sources to collect data. Denote the corresponding total queue differential as $\sum_{i,j} w(A_3(i,j))$.

In the third epoch, the $N-K$ robots move to corresponding sinks to deliver data and the K robots move to sources to collect data. At the end, the $N-K$ robots finish delivery and their queue become empty. The total queue differential between robots and sinks associated with their served sources is $\sum_{i,j} w(A_3(i,j)) + N\lambda T$ (or a little bit less if there are leftovers exist in the sources, but this will not affect the allocation except for driving the $N-K$ robots to serve sources with more leftovers), which is the maximum total queue differential. This is because at the beginning of the third epoch, the total queue differential $\sum_{i,j} w(A_3'(i,j))$ provided by any other allocation A_3' is no greater than $\sum_{i,j} w(A_3(i,j))$. And during the third epoch, each source an amount of λT new data arrive, and at most N sources can have their new data arrived in the third epoch completely being collected by their serving robots. Thus, at the beginning of the fourth epoch, any allocation cannot provide a total queue differential greater than $\sum_{i,j} w(A_3(i,j)) + N\lambda T$. Thus, in the fourth epoch, based on the CBMF, the allocation that solves the optimization problem (P1) is to assign robots to sinks associated with the sources they serve in the third epoch.

As we apply the above analysis for all epochs, it can be proved that robots are divided into two groups whose size are K and $N-K$ separately. And the two groups interchangeably serve sources and sinks in alternative cycles. Similarly, since all flows are homogeneous, each flow has an equivalent amount of time being served. Thus, the robotic allocation follows the Robot Allocation Strategy II shown in Algorithm 4.