Routing Protocol Design for Drone-Cell Communication Networks

Peng Yang*‡§, Xianbin Cao*‡§, Chao Yin*‡§, Zhenyu Xiao*‡§, Xing Xi*‡§ and Dapeng Wu†*
*School of Electronic and Information Engineering, Beihang University, Beijing, China
†Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL 32611, USA
‡Beijing Laboratory for General Aviation Technology, Beihang University, Beijing, China
§Collaborative Innovation Center of Geospatial Technology, Wuhan 430079, China

***Abstract*—This paper is concerned with the design of routing protocol capable of congestion mitigation for drone-cells communication networks where drone-cells remain stationary in the sky as relays. All of the (distance or hop-count based) existing routing protocols can perform well when the network is lightly loaded. Once the network is heavily loaded, a large number of packets might be backlogged in queues of network nodes since these protocols can not be aware of the network congestion condition. In this paper, we propose a queuing delay and transmission delay based routing protocol (QDTD) to relieve the network congestion caused by heavily loaded traffic. First, QDTD designs a novel ForWard-Back (FWB) queue architecture that significantly reduces the number of queues maintained at each network node. Second, both queuing delay and transmission delay are leveraged as a routing metric to enhance the performance of QDTD. Experimental results show that QDTD can effectively relieve the network congestion and reduce the overall network delay and achieve high throughput.**

***Index terms* Routing protocol, drone-cells, congestion mitigation, forward-back queue**

I. Introduction

Transportation and communication technologies have been changing our lifestyle greatly. Combining the state-of-the-art advancements in these two technologies, drone-cells, i.e., drones equipped with transceivers, assisted wireless communication networks are being actively researched and developed as a key solution to boost the coverage and enhance the quality of service (QoS) of existing ground heterogeneous networks. Moreover, drone-cells assisted wireless communication networks' networking-transportation-communication are expected to boom in emerging civilian applications such as search and rescue missions [1], reconnaissance over disaster recovery [2], and remote sensing [3]. However, it is particularly challenging to design effective communication mechanisms for drone-cells networks due to the following reasons. First, the wired networks composed of electrical cables and optical links or wireless cellular networks on the ground can support the demand for high-speed data rates. Compared to the ground networks, the capacity and computing power of drone-cells networks are rather limited; thus, the continuous growth data request poses a great challenge to drone-cells networks. Second, since drone-cells are leveraged to provide radio access network (RAN) elements (supply) for flash crowd traffic demands, the distribution of which in both time and space is random, the network congestion condition may be

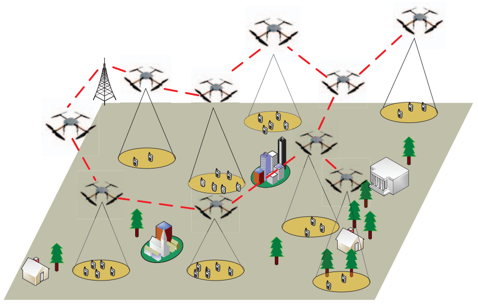
Fig. 1. Drone-cell network.

common in drone-cells networks, which leads to a significant amount of backlog data in drone-cells' queues. What's more, a large backlog of data might increase the overall network delay. Even though we can assume that the queue size in a drone is massive, the usage of a large capacity memory will enlarge the drone-cell cost and can not prevent the occurrence of the queue backlog. Therefore, it is expedient to utilize the available capacity fully in drone-cells networks by designing efficient network control algorithms.

To prepare for the launch of a drone-cells assisted communication network, this question is required to be addressed: what routing protocol is suitable for drone-cells communication networks under any feasible load? We formulate this problem as the following: this paper considers a particular case of multiple terminals and a single destination base station that is depicted in Fig. 1. When a large number of communication infrastructures in a ground area are destroyed by a natural disaster, for example, an earthquake, many congested terminals on the ground can only access the outside internet through relay nodes consisting of quadratic drone-cells that stay stationary in the sky. What routing protocol can be employed to relieve the network congestion and achieve small network delay and high throughput when ground terminals create large volumes of data requests?

Network congestion has been studied extensively in the wireless network filed during the past few decades. Transmission control protocol (TCP) is a famous transport layer protocol that can support the network congestion control [4]. There are several different flavors of TCP congestion control, each of which operates somewhat differently [5-7].

However, all versions of TCP are window-based protocols. The idea is that each source node manages a value, called the window size, which is the number of unacknowledged packets allowed to transmit into the network. Any new packet can be transmitted only when the sender receives the ACK for one of the previously sent packets. TCP adapts the window size in response to congestion condition. In essence, TCP is to solve the problem of utility maximizing in a network, and there is a critical assumption that the capacity constraints of the links are time-unvarying in this optimization problem. However, due either to fading channels or changing connectivity because of mobility, the link capacity is always time-varying in a wireless network. Furthermore, directly applying TCP may incur severe unfairness in resource usage among competing traffic when the traffic load of the wireless network is high [8].

To solve the congestion control problem of the wireless network, the seminal work of [9] introduced a joint adaptive routing and scheduling method, called back pressure, which had been shown to be throughput-optimal, i.e., it could stabilize a network under any available load. The back pressure method calculated the link weight as the product of the link capacity and the maximum "back pressure" (i.e., the queue length difference between the queues at the upstream node and downstream node of this link for each flow) among all the flows passing through this link. Then it solved a MaxWeight programming problem to activate a set of non-interfering links that had the largest weight summation. The flow with the maximum queue length difference at a link was then selected to transmit packets when this link was activated. Besides, a weighted back pressure method was proposed in [10], and a shortest path bias was employed to enhance this back pressure method so that in low loading situations, nodes were inclined to route packets in the direction of their destinations. However, the above back pressure methods are based on the per-destination (PD) queue. The PD queueing system requires each node to manage a queue for a flow; thus, the number of the queues maintained by each node will equal to the total number of flows throughout the network. Moreover, the PD based back pressure method exists many drawbacks: 1) it needs per-destination or per-flow information that is usually difficult to obtain and manage, especially in large networks where there are numerous flows; 2) it requires to separately manage a queue for each destination or flow at each node; 3) it may lead to poor overall network delay performance because the queue length requires to be built up (creating the back-pressure) from a flow destination to its source, which results in massive queues along the path a flow takes [11,12].

To address the shortcomings of PD based back pressure method, a TRiple-node Indexed (TRI) queue based back pressure method, trickle, was proposed recently [13]. Trickle provided a similar functionality to the conventional PD queues but with a more lightweight and practical design. In the Trickle queueing system, each node managed multiple TRI queues for each of its (parent, node, child) flow pairs; thus, it required each node to maintain at most n^2 queues, where n was the number of neighbor nodes. Although trickle is throughput optimal and helps with the network congestion control, it may lead to large end-to-end network delay. According to the trickle, nodes will incline to route packets in the direction of decreasing queue backlog. When the network is in low loading, packets may take many false turns resulting in a significant delay in large networks.

In this paper, we design a novel ForWard-Back (FWB) queue model that requires each network node to maintain at most n queues. We further propose a queuing delay and transmission delay based routing protocol (*QDTD*) capable of alleviating the network congestion conditions. In *QDTD*, a queuing delay and a transmission delay are combined as a routing metric. We conduct an experiment to evaluate the performance of *QDTD*. Experimental results demonstrate that *QDTD* can achieve better performance than the hop-count based routing scheme.

Note that our work may be similar to [10] in that we also design a minimum cost metric to route packets to their destination. However, the differences between the two are as follows: 1) we employ the FWB queue backlog based and minimum hop-counts based delay information to route packets other than utilize the sum of a differential PD queue backlog value and a minimum distance. 2) [10] only describes an algorithm without implementation, performance evaluation, and comparison with other algorithms.

The rest of the paper is organized as the following. Section II presents a detailed description of the system model. Section III gives an overview of the proposed protocol. In Section IV, we describe our proposed *QDTD* framework in details. In Section V, we show the experimental results. Section VI concludes this paper.

II. System model

In this section, we will present the system model of *QDTD*. To better illustrate our system, we use the word "node" to represent a drone-cell. First, we consider a wireless network modeled by a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is a set of nodes and $\mathcal{L}$ is a set of links. A link (i, j) denoted by l_{ij} exists if it is in $\mathcal{L}$. The neighbor nodes collection of a node i is denoted by N_i. We assume a slotted system, and the time slot is denoted by t. In every time slot, let $s_{ij}(t)$ represent the instantaneous channel state of $l_{ij} \in \mathcal{L}$, where i and j are the transmitter and receiver of the link. The channel state matrix of the network can be denoted by $S(t) = [s_{ij}(t)]$. Every time slot t, a network controller will determine transmission rate on each link by allocating a power matrix $P(t) \in \Pi$ where Π is a compact set of acceptable power allocations.

Let $\mathcal{F}$ be the set of flows consisted in the network where each of them is indexed by $f = 1, 2, ..., |\mathcal{F}|$ and $|\cdot|$ is the cardinality of a set. The destination node of flow f is denoted by f_d. Let u_l represent the transmission rate of the link l, and then a schedule $\pi(t) = (u_1^{\pi(t)}, u_2^{\pi(t)}, ..., u_{|\mathcal{L}|}^{\pi(t)})$ is the link rates that can be supported simultaneously by the network.

Consider a single queue with an input process $A(t)$ and a transmission rate process $u(t)$, where $A(t)$ represents the amount of new arrivals that enter the queue during slot t, and

$u(t)$ denotes the transmission rate of the server during slot t. We assume that the $A(t)$ arrivals occur at the end of slot t so that they cannot be transmitted during that slot. Let $U(t)$ represent the current backlog in the queue. The $U(t)$ process evolves according to the following discrete time queueing law

$$U(t+1) = [\underset{\text{当前储量}}{U(t)} - \overset{\text{发送掉的量}}{u(t)}]^+ + \underset{\text{进入流量}}{A(t)} \tag{1}$$

where, $[t]^+ = max(t, 0)$.

The queue might be located within a larger network, in which case the arrival process $A(t)$ is composed of random exogenous arrivals as well as endogenous arrivals resulting from routing and transmission decisions from other nodes of the network. Likewise, the transmission rate $u(t)$ can be determined by a combination of random channel state variations and controlled network resource allocations, both of which can change from slot to slot.

We assume that the necessary condition of a ground user being in the coverage region of a drone-cell is that the air-to-ground (ATG) link satisfies its QoS requirement. The ATG link model that considers the line-of-sight (LOS) signal and non-line-of-sight (NLOS) signal along with their occurrence probabilities can be found in [14]. Furthermore, the free space propagation model is employed to model the air-to-air (ATA) propagation channel in this paper. Thus, the expression of pathloss between any two drone-cells can take the form [14]

$$L(r) = 20\log\left(\frac{4\pi f_c}{c}\right) + 20\log(r) \tag{2}$$

where, f_c (in Hz) is the carrier frequency, c (in m/s) is the speed of light, and r represents the distance between two drone-cells.

III. System Overview

In this section, we will give an overview of the *QDTD* that is designed for drone-cells communication networks under any available load.

In *QDTD*, once stationary quadratic drone-cells receive data packets from ground terminals or other drone-cells, they act as relay nodes and constantly seek a proper next hop to hand over stored packets. This process goes repeatedly until the packets reach the destination.

In our FWB queue structure, each node needs to maintain a set of "forward queues" and a set of "back queues". A "forward queue" is employed to hold data packets forwarded to a next hop. A "back queue", however, is involved in recording the number of data packets received from a previous node and forwarded by the current node. In this case, we can utilize counters to realize the counting function of "back queues". Therefore, each node only requires maintaining n queues with n representing the number of neighbor nodes.

With the usage of FWB queues, *QDTD* can effectively perceive the network congestion conditions by calculating the FWB queue backlog difference. To lower the network delay and relieve the network congestion, *QDTD* adopts an enhanced routing metric consisting of a *queuing delay* and a *transmission delay*. With the routing metric, nodes are inclined to route packets in the direction of the destination node while alleviating the network congestion.

Since the calculation of the routing metric requires the exchange of the FWB queue backlog and the minimum hop-count information between neighbor nodes, the promiscuous mode of each node is enabled to reduce the overload of information exchange. Before transmitting a packet, each node will piggyback its FWB queue backlog information and minimum hop-count information and some other information in the *QDTD* header of a packet. Then its neighbor nodes can obtain the needed information through eavesdropping and parsing data packets.

IV. Description of *QDTD*

As mentioned in previous sections, the *QDTD* framework aims to design a routing protocol capable of congestion control so that the network delay is reduced and the throughput is improved. In this section, we present the detailed description of how and why *QDTD* works.

A. *Forward-Back Queue Model*

Definition 1. Virtual Source Node (VSN). Given a link (i, j) in the network, node i will transmit packets to node j. If the transmitted packets consist of two components, endogenous (or transfer) packets, and exogenous (or local) packets, we then say that the node i exists a mapping node, called the virtual source node, which generates equivalent exogenous packets for the node i.

Regarding this definition, there are a few points to explain: 1) if a node does not generate the local traffic, there will be no corresponding VSN; 2) when a VSN transmits local packets to the corresponding original node, the original node will act as a relay to forward the received packets.

The input and output queue models for FWB queue model are illustrated in Fig. 2 and Fig. 3 respectively. Solid lines denote links between two nodes and dash lines represent the route between two queues to transfer data. Let $\vec{u}_{i,j}(t)$ denote the input rate of the link l_{ij} at time t, "forward queue" $q_{i,j}$ indicate the queue holding data from i to j, and "back queue" $v_{i,j}$ represent a virtual queue, which can be implemented by a counter, holding data in forward node j with node i as the previous hop node. For a packet in $q_{i,j}$, when it is transmitted to node j, the destination of it will be checked against j. If the packet is destined to node j, it will disappear from the network; otherwise, it will enter another queue, for example, $q_{j,k}$ in agreement with the routing. Next, the packets not destined to node j yield a data rate from $q_{i,j}$ to $q_{j,k}$. Let $\overset{\text{q\_ij的出速率}}{u_{i,j}(t)}$ represent the allocated output data rate of $q_{i,j}$ and $u_{i,j,k}(t)$ represent the allocated data rate from $q_{i,j}$ to $q_{j,k}$. Additionally, we let $u_{i,i,j}$ represent the compound allocated data rate going through the virtual source node i, the original node i and node j.

q_ij的入速率 (i,j)这条边的input rate

q_ij在i端, 存储了i打算传给j的信息

v_ij在j端, 存储了i传给j的信息

如果这个packet是传给j的, 那么它将消失, 否则, 它将会进入下一个forward queue

queue ij转给queue jk的信息量

Then, for a link (i, j), its input and output rate can take the following forms

$$\underset{\text{i的所有邻居给q\_ij的贡献}}{\vec{u}_{i,j}(t)} = \underset{\text{i的邻居并上它自己}}{\sum_{m\in\{N_i,i\}}} \overset{\text{q\_mi -> q\_ij的量}}{u_{m,i,j}(t)} \tag{3}$$

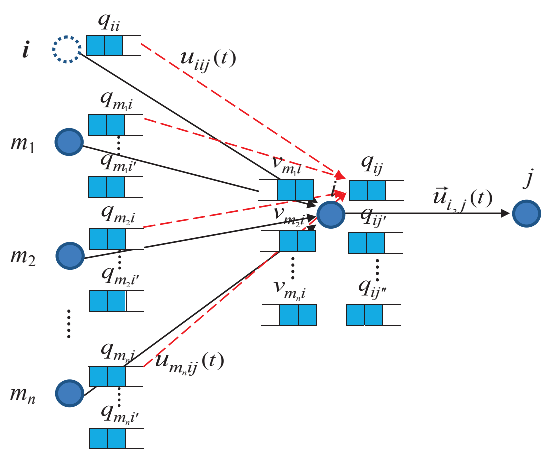

Fig. 2. The input model for FWB queue.

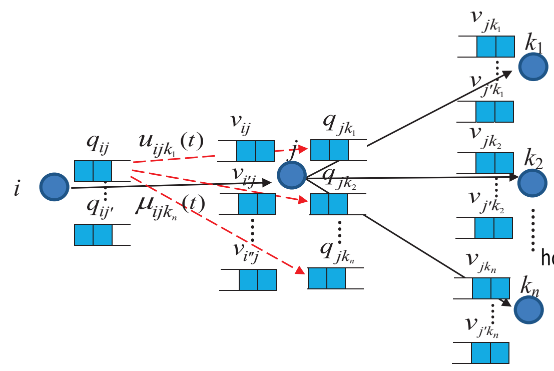

Fig. 3. The output model for FWB queue.

q_ij转到q_jk.

$$\overleftarrow{u}_{i,j}(t) = \sum_{k\in N_j} u_{i,j,k}(t) \qquad (4)$$

q_ij的信息有多少被j转发

where N_i represents the set of neighbor nodes of the node i.

Since each queue contains traffic for different flows, after regarding the flow rates we can write $u_{i,j,k}(t)$ in this form

$$u_{i,j,k}(t) = \sum_{f:f_d\neq j} u^f_{i,j,k}(t) \qquad (5)$$

j可以决定给q_ij里的信息有多少分到q_jk

where $u^f_{i,j,k}(t)$ is the allocated rate for flow f on l_{ij} from $q_{i,j}$ to $q_{j,k}$ at time t, f_d represents the destination node of flow f.

Let $q_{i,j,k}$ denote a queue that tracks traffic from $q_{i,j}$ to $q_{j,k}$. Then the queue backlog of $q_{i,j,k}$ at time $t+1$ can be updated according to this form

j那里堆积的i到k信息 j收取i的速率

$$q_{i,j,k}(t+1) \le \left[q_{i,j,k}(t) - \overleftarrow{u}_{j,k}(t)\right]^+ + \overrightarrow{u}_{i,j}(t) \qquad (6)$$

k收取j的速率

It is an inequality instead of equality because the arrivals may be less than the allocated output data if neighbor nodes have little or no data to transmit [10].

With (6), the queue backlogs of "forward queue" $q_{i,j}$ and "back queue" $v_{i,j}$ can be respectively written as

i那里堆积的到j的信息

$$q_{i,j}(t) = \sum_{m\in N_i} q_{m,i,j}(t) \qquad (7)$$

$$v_{i,j}(t) = \sum_{k\in N_j} q_{i,j,k}(t) \qquad (8)$$

j那里堆积的i到k的信息

B. Routing Metric and Decision

To reduce the network delay, we can employ a delay based routing metric to route packets to their destination, for example, the *transmission delay* representing the time of forwarding a packet through the path with minimum hop-counts. However, if we only use this metric for the network under heavy load, a large number of data will be blocked at the minimum hop-counts path incurring big packet delay. Therefore, we program a *queuing delay*, which denotes the time of transferring all the differential FWB queue backlog packets, into the *transmission delay* metric in this paper. By considering the *queuing delay*, packets will be routed in the direction of other nodes with lower queue backlog so that the network load is balanced and the network congestion is relieved when the network traffic is heavy. Then we can summarize the *QDTD* routing protocol as the following.

目前的metric主要是时延

有什么问题

我们提出了queue delay

packet将会被发给lower queue backlog的节点

Algorithm 1 *QDTD* routing protocol

Purpose: Next Hop Decision

1: Routing metric: Denote the routing metric or weight of the link (i,j) by $W_{i,j}$

这块写的不太对，问了师兄，他说u不应该在分子的,不然u越大W越小

packet size i给j实发 j实际收到 B: 到destination的距离

$$W_{i,j}(t) = \frac{L}{u_{i,j}(t)}\left(q_{i,j}(t) - v_{i,j}(t)\right) + \frac{(B_i(t) - B_j(t))\times L}{u_{i,j}(t)} \qquad (9)$$

hop评价函数，越大越好 transmission rate

where $u_{i,j}(t)$ is the transmission rate on the link (i,j) at time slot t, L represents the size of a packet, and B_i denotes the minimum number of hop-counts between the node i and the destination.

2: For all links (i,j), find the suitable next hop (denoted by j^*) for node i, such that 希望 u 较大(信道质量好), queue较长(信息积压多),到重点hop数缩短的多的点W越大

选这个作为i的next hop

$$j^* = \arg\max_{j\in N_i} W_{i,j}(t) \qquad (10)$$

i的邻居

3: Routing Decision: define transmission rates as follows

power allocation

$$u_{i,j}(t) = \begin{cases} u_{i,j^*}(P(t), S(t)) & if \quad j = j^*\, and\, W_{i,j^*}(t) > 0 \\ 0 & otherwise \end{cases} \qquad (11)$$

channel state

where $P(t) = P_t$, if the link (i,j) is activated; otherwise, $P(t) = 0$.

Note that the $W_{i,j^*}(t)$ value represents the maximum value of the summation of the *queuing delay* and the *transmission delay*. With this new routing metric, the proposed algorithm does not need any pre-specified set of routes, i.e., the path for each packet can be created dynamically.

C. Implementation of QDTD

An essential part of implementing *QDTD* is how to exchange hop-count and queue length information among neighbor nodes. To save the overhead of retrieving this information, we enable the node's promiscuous function, where each node can eavesdrop its neighbor nodes for any required information. Therefore, before a node transmits a data packet, we add an extra *QDTD* header before the IP header in a data packet. The packet structure is illustrated in Fig. 4. The 32-bit field *Destination address* represents the IP addresses of

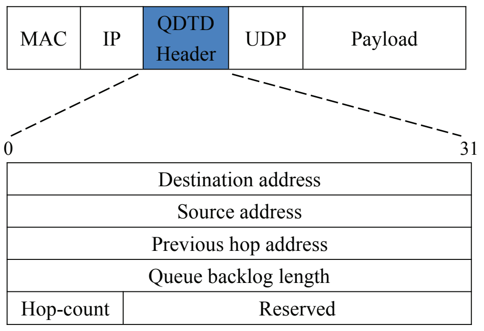

Fig. 4. *QDTD* packet structure.

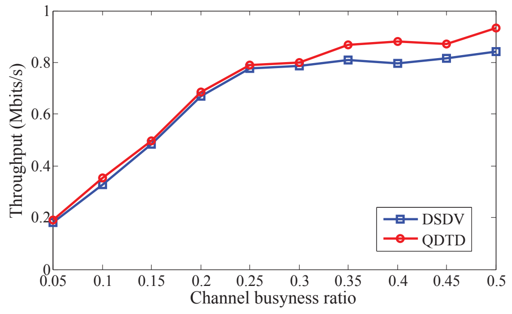

Fig. 5. Illustration of the average throughput on different channel busyness conditions for *DSDV* and *QDTD*.

the destination node of this packet. Field *Source address* is the IP address of the source node of this packet. This field *Previous hop address* is the IP address of the node that the last received packet comes. *Queue backlog length* field represents the queue backlog length information the transmitting packet piggybacks. It is calculated at the current sending node before the packet is transmitted using (8). *Hop-count* field represents the minimum hop-counts between the sending node and the destination node.

V. Experimental Results

A. Development of Simulator

First, we implement the shortest path based routing protocol, *DSDV*, which takes the minimum hop-count between the current node and the destination as the routing metric. We also implement our proposed *QDTD* routing protocol. These two protocols are carried out with Linux Ubuntu 16.04 LTS and on the NS-3 simulator.

We consider the special case of a "multiple sources single destination" static square network with 24 quadratic drone-cells and one base station locations discretized to a grid. This network is employed to supply communication connection for ground users in a $2000 \times 2000m^2$ urban area. In this paper, we focus on discussing the backhaul communication network consisting of the drone-cells and the base station. The base station as the destination is located at the upper left corner of the grid and four source drone-cells, which can collect data from ground terminals, are located at the other three corners and the center location, respectively.

We use UDP as the Layer 4 protocol. In the experiment, we have the following parameter setting: the packet size L = 1500 bytes, the queue size is 600 kilobytes. The transmission power of a drone-cell $P_t = 17dBm$, the carrier frequency $f_c = 5.15GHz$, noise power $P_N = -33dBm$, the pathloss corresponding to the ATA link QoS requirement $\gamma_d = 108dB$. The pathloss corresponding to the ATG link QoS requirement $\gamma_u = 105dB$, and the altitude of a drone-cell is 141m.

B. Performance Evaluation

In the following experiment, we study the effects of channel busyness ratio on the end-to-end average throughput and the average network delay. We choose different random seeds and conduct experiments ten times and take the average value.

For the experimental result of end-to-end network average throughput, we only keep track of the data packets and ignore the control packets and measure the throughput in Mbits/s under different values of channel busyness ratio. In a distributed drone-cells network as we study, it is difficult to adjust the channel busyness ratio to our required value appropriately. Hence, we adopt a simplified scheme that uses the maximum channel busyness ratio of a single node in the network. Channel busyness ratio is defined as the fraction of a constant time unit (e.g., one second) in which the channel is in the non-idle state. This includes the channel either being occupied by transmission of particular data or in such a working state that it can not be shared [15]. Since the channel congestion condition is mainly determined by the transmission rate, we can achieve different channel busyness ratio by adjusting the transmission rate of ground source nodes. However, we find that when the transmission rate of source nodes increases to a certain level, there is a bottleneck that saturates the channel busyness ratio (e.g., on 50%).

Fig. 5 illustrates the resulting end-to-end average throughput vs. channel busyness ratio of the two comparison routing protocols, *DSDV*, and *QDTD*. From this figure, we can achieve the following observations:

- When the channel busyness ratio is less than 20%, that is, the network is in light congestion, the throughput values of both *QDTD* and *DSDV* routing protocols will rise quickly with the increasing of the channel busyness ratio. Furthermore, *QDTD* achieves slightly higher throughput than that of the *DSDV*. The reason is as follows: for *QDTD*, it adopts both the *queuing delay* and the *transmission delay* information to route packets. Then, packets can be routed via the "Routing-Bundle" that consists of the path with minimum hop-counts and multiple paths clustering around it to reach the destination.
- When the channel busyness ratio is between 20% and 30%, i.e., the network congestion is moderate, things are different. The growth rate of the throughput values achieved by both *QDTD* and *DSDV* are gradually slowing down with the increase of transmission speed.
- When the channel busyness ratio is below 30%, the

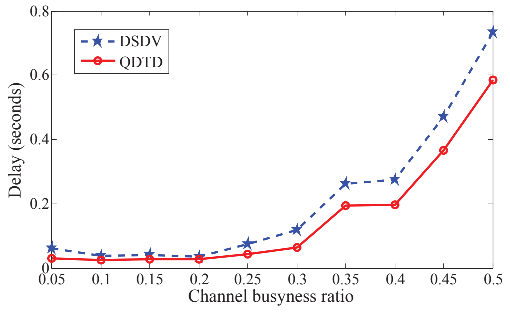

Fig. 6. Illustration of the average packet delay on different channel busyness conditions for *DSDV* and *QDTD*.

resulting throughput trends for both *QDTD* and *DSDV* are close. This is because the *transmission delay* dominates the routing decisions when facing no severe network congestion conditions. However, the *queuing delay* also leads to a limited throughput improvement.

- When the channel busyness ratio is higher than 30%, the throughput achieved by *DSDV* no longer increases. This is because the *DSDV* can not be aware of the network congestion conditions. However, due to the existence of the effective congestion control mechanism the resulting throughput gained by *QDTD* is boosted.

Fig. 6 illustrates the resulting average delay vs. channel busyness ratio of the two routing protocols. From the figure, we can have the following observations:

- Thanks to the principle of the shortest path, both *DSDV* and *QDTD* can route a packet to the destination with small network delay when the network is in light congestion. Moreover, *QDTD* can route packets with lower delay. As we stated above, *QDTD* can route packets through the "Routing-Bundle"; thus, the waiting time for packets to be sent is shortened.
- The situation will become different when the network congestion is moderate. Because *DSDV* will continuously adopt the path with minimum hop-counts to deliver packets, packets have to take a long time to wait for being sent. Regarding the *QDTD*, it can maintain a small network delay until the channel busyness ratio reaches 30%. This is because the *queuing delay* in *QDTD* obtains a little effect.
- When the channel busyness ratio exceeds 30%, the network becomes more congested; thus, the network delays of both *QDTD* and *DSDV* rise rapidly. However, due to the significant effect of the congestion control *QDTD* can still route packets with a smaller delay than *DSDV*.

In summary, experimental results demonstrate that *QDTD* achieves higher throughput and lower network delay than *DSDV* in drone-cells networks under any feasible load.

VI. Conclusion

This paper was concerned with routing protocol design of drone-cell communication networks under any feasible load. To alleviate network congestion, a queuing delay and a transmission delay based routing protocol, *QDTD*, was proposed. *QDTD* designed a novel lightweight ForWard-Back (FWB) queue architecture. By leveraging the queuing delay and the transmission delay as a routing metric *QDTD* could adaptively determine an appropriate path for packets to reach the destination while incurring small network delay, achieving high throughput, and relieving network congestion. Experimental results demonstrated that *QDTD* was effective. In our future work, we will investigate rate control at end hosts/terminals to further improve the congestion control performance.

Acknowledgment

This paper is supported by the National Natural Science Foundation of China (Grant No. 91538204), the National Science Fund for Distinguished Young Scholars (Grant No. 61425014) and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61521091).

References

[1] J. Manathara, P. B. Sujit and R. Beard, "Multiple UAV coalitions for a search and prosecute mission," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 1, pp. 125-158, 2011.
[2] I. Maza, F. Caballero, J. Capitan, J. R. Martinez-De-Dios, and A. Ollero, "Experimental results in multi-UAV coordination for disaster management and civil security applications," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 563-585, 2011.
[3] I. Boryaliniz and H. Yanikomeroglu, "The new frontier in RAN heterogeneity: Multi-tier drone-cells," *arXiv: 1604.00381v2*, 2016
[4] J. Postel, "RFC 793: Transmission control protocol," *Internet Request for Comment*, vol. 2, no. 4, pp. 595-599, 1981.
[5] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Symposium on Communications Architectures and Protocols*, ACM, 1988, pp. 314-329.
[6] V. Jacobson, "Modified TCP congestion avoidance algorithm," *Technical report*, 1990.
[7] S. Floyd, T. Henderson and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," *RFC 3782*, 2004.
[8] A. Warrier, S. Janakiraman, S. Ha and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," in *IEEE INFOCOM 2009*, pp. 262-270.
[9] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, 1992.
[10] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89-103, 2003.
[11] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1597-1609, 2011.
[12] A. Stolyar, "Large number of queues in tandem: Scaling properties under back-pressure algorithm," *Queueing Systems*, vol. 67, no. 2, pp. 111-126, 2011.
[13] Q. Y. Huang, J. D. Li, Y. Zhu and D. P. Wu, "Trickle irrigation: congestion relief for communication with network coding," *submitted to IEEE/ACM Transactions on Networking*.
[14] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569-572, 2014.
[15] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer, "Understanding congestion in IEEE 802.11b wireless networks," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. ACM, 2005, pp. 279-292.