RUSH: RoUting and Scheduling for Hybrid Data Center Networks

Kai Han*† Zhiming Hu* Jun Luo* Liu Xiang‡

*School of Computer Engineering, Nanyang Technological University, Singapore
†School of Computer Science and Technology, University of Science and Technology of China, China
‡Institute for Infocomm Research (I2R), A*STAR, Singapore
Email: hankai@gmail.com; {zhu007,junluo}@ntu.edu.sg; xiangl@i2r.a-star.edu.sg

***Abstract*—The recent development of 60GHz technology has made *hybrid Data Center Networks* (hybrid DCNs) possible, i.e., augmenting wired DCNs with highly directional 60GHz wireless links to provide flexible network connectivity. Although a few recent proposals have demonstrated the feasibility of this hybrid design, it still remains an open problem how to route DCN traffics with guaranteed performance under a hybrid DCN environment. In this paper, we make the first attempt to tackle this challenge, and propose the RUSH framework to minimize the network congestion in hybrid DCNs, by jointly routing flows and scheduling wireless (directional) antennas. Though the problem is shown to be NP-hard, the RUSH algorithms offer guaranteed performance bounds. Our algorithms are able to handle both batched arrivals and sequential arrivals of flow demands, and the theoretical analysis shows that they achieve competitive ratios of $\mathcal{O}(\log n)$, where n is the number of switches in the network. We also conduct extensive simulations using *ns*-3 to verify the effectiveness of RUSH. The results demonstrate that RUSH produces nearly optimal performance and significantly outperforms the current practice and a simple greedy heuristics.**

I. Introduction

With the proliferation of cloud computing as an on-demand network service, the supporting infrastructure, data centers, has attracted great attentions. In particular, as tens to hundreds of thousands of servers may potentially work together in a data center and coordinate with each other through underlying networks, the *Data Center Network* (DCN) becomes the core component of offering qualified cloud computing services. Improving the DCN performance is one of the primary research issues for the networking community [1]–[5]. Although interconnecting co-located servers may appear to be trivial given the extensive deployments of local area networks, the huge number of servers connected by a DCN and the tremendous amount of network traffic they produce make the construction of efficient but low cost DCNs a challenging problem.

While early DCNs apply tree-structured topologies involving Ethernet switches for low-cost interconnections of a relatively small number of servers, the growth in the scale of DCNs pose very heavy load on some links in the tree topology (oversubscription by a factor of 2.5 to 8 is possible [1]). More recent research efforts tend to introduce complicated topologies to address the traffic concentration problem [1]–[3]. While FatTree [1] and VL2 [2] both follow the Clos-type of topology but differ in whether a few faster (non-commodity) switches are used, BCube [3] innovates in abandoning the hierarchal structure and using hosts to relay traffic, which results in a hypercube-like topology.

While all the wired topologies offer a large range of choices to balance between the needs of efficiency and low cost, their flexibility is always a questionable issue (besides other drawbacks such as unnecessary cost to suppress oversubscription) [4], [5]. More specifically, modifying the topology of an already deployed DCN is very complex and may incur high cost, due to the multi-staged design and the physical constraints on arranging the wires (e.g., the bundled feature of fibers). Consequently, several recent research proposals have started to explore the possibility of using wireless networks (operating in the 60GHz band) to offload the traffic in wired networks [4]–[6]. The flexibility of having a hybrid DCN is evident: the wireless links can be established in an on-demand manner that exactly suits the on-demand feature of the supported cloud computing services. While the relatively theoretical work in [6] applies a genetic algorithm to perform channel allocations to wireless links in a hybrid DCN, the practical deployments demonstrated in [4], [5] mainly aim to confirm the feasibility of hybrid DCNs with directional antennas, without actually taking care of the optimal flow scheduling issue. As optimally scheduling flows (hence the antenna directions for wireless links) is crucial for improving the DCN performance, **a relevant question now is whether we can perform joint (flow) routing and (antenna) scheduling for hybrid DCNs with directional antennas while providing a certain level of guarantee on performance.**

To answer this question, we present in this paper RUSH as a practical optimization framework for fast or online flow scheduling in hybrid DCNs. We inherit the basic setting of a hybrid DCN from [4]. Briefly, the *base wired network* follows a hierarchical topology[1] and is provisioned for average case. In addition, each *Top-of-Rack* (ToR) switch is equipped with one 60GHz wireless device. Also, a central scheduler is available to monitor the traffic and schedule both flow routes and wireless antenna directions. RUSH bears an optimization

[1]RUSH is rather independent of the wired topology, so one may potentially choose any type of topology available.

objective of minimizing the maximum congestion level in all links, which is a desirable feature for DCNs [4]. To achieve the objective, our RUSH framework includes two optimizers: RUSH-Batch for handling batched arrivals of the (flow) routing requests and RUSH-Online to deal with sequential arrivals in an online manner. Both algorithms have a provable competitive ratio of $\mathcal{O}(\log n)$, where n is the total number of switches deployed in a hybrid DCN. They also entail low computation complexity and can hence be performed in a fast or online manner.[2] We perform extensive simulations with *ns*-3 to confirm the effectiveness of the RUSH algorithms.

The remaining of our paper is organized as follows. We discuss the background and system models in Sec. II, and we also define the optimization objective and formulate the optimization problem in the same section. Then we present the two RUSH algorithms in Sec. III and Sec. IV, respectively. We also report the results of our numerical computations and simulations in Sec. V. We finally discuss the relate work in Sec. VI, before concluding our paper in Sec. VII.

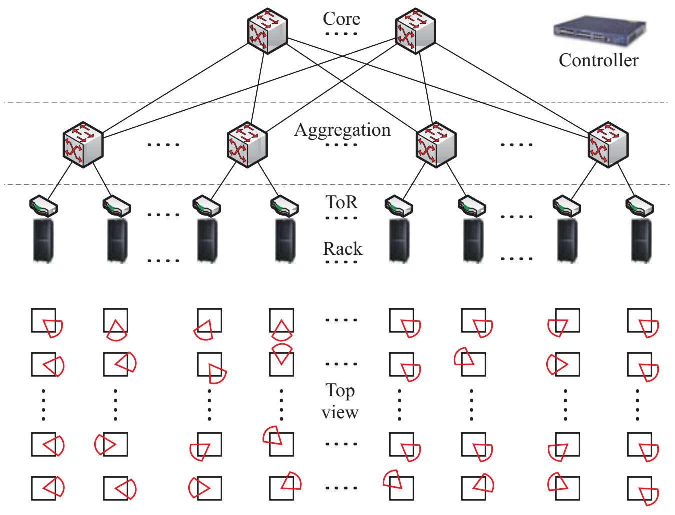


Fig. 1. A hybrid data center network. The red sectors represent the directional antennas of the wireless devices co-located with ToRs.

II. Background, Models and Problem Formulations

In this section, we first briefly describe the design of hybrid DCNs, then we present our model for the 60GHz wireless links used in hybrid DCNs. Finally, we define our optimization objectives and formulate our problem.

A. Hybrid DCNs Overview and Network Model

We use Fig. 1 to illustrate the design of a hybrid DCN. The upper part of Fig. 1 shows a common 3-stage multi-rooted tree topology for the wired DCN, while the lower part emphasizes (through a top view) that each ToR is equipped with a 60GHz direction antenna. Although our RUSH framework is independent of the wired topology, we prefer this simple topology over other more advanced designs [1]–[3] for the ease of exposition. We only consider a DCN at the granularity of ToR, as the links among servers under each ToR (belonging to the same rack) are rarely saturated [7]. Therefore, we will model a DCN as a set of n nodes, each corresponding to a switch (it can be a ToR, aggregation, or core switch), where traffic may enter or leave only at ToR switches. We assume that a central controller exists to coordinate the network traffic; this can be achieved by an extension of OpenFlow-enabled switches [8]. Basically, each switch maintains statistics about flows passing through, and the controller can query existing flow entries, perform optimization based on our RUSH, and schedule paths for each arriving flows.

Almost all the recent proposals on hybrid DCNs suggest using 60GHz wireless technologies, mainly because they offer the advantage of larger bandwidth (data rate) and lower interference (due to beamforming). Also, radios that operate in 60GHz unlicensed band are available, with directional antennas that can be steered mechanically or electronically [4], [5]. Therefore, we also assume 60GHz wireless links for our hybrid DCN design. Our RUSH framework aims to schedule flows for minimizing the congestion, which in turn implies steering directional antennas to select proper wireless links that can offload the congested wired links.

Combining the wired and wireless part, we model a hybrid DCN as a graph $G = (V, E)$, where V is the set of n nodes and E is the set of communication links. The set V consists of two disjoint subsets: V_R includes all the low tier (or ToR) switches and V_H contains the other switches. According to our discussion in Sec II-A, flows may start or end only at V_R. The links in E are also divided into two disjoint subsets E_D and E_W: E_D is the set of wired links and E_W is the set of wireless links. As only nodes in V_R can operate wireless links, we also call these nodes *wireless nodes*. For any link $e \in E$, the capacity of e is denoted by $c(e)$. The set of all wireless links incident to a node $v \in V_R$ is denoted by $\partial(v)$.

As the nominal capacity of a single 60GHz wireless link can be very high in an indoor environment ($\approx$ 6Gbps according to [4]), we can afford to run it at a much lower rate (e.g., 1Gbps) that makes individual links very robust against SINR interferences, while we conservatively assume that all other interfering links are transmitting concurrently. Actually, such a conservative SINR interference (rate selection) model is also adopted in [4], [9], and its feasibility is further corroborated by the highly directional antennas and the newly devised 3D beamforming approach [5], as well as the availability of three orthogonal channel under the 60GHz band (which can be allocated to reduce interference through simple heuristics given the regular node deployment in a hybrid DCN).

B. Joint Routing and Scheduling for Hybrid DCNs

We assume that there are a set of routing requests $R = \{r_1, r_2, ..., r_K\}$ injected into a hybrid DCN G through V_R, where $r_i = (s_i, d_i, b_i)$ and s_i, d_i, b_i are the source node, the destination node and the traffic demand of r_i, $\forall 1 \leq i \leq K$, respectively. Each request is followed by a flow (a TCP

[2]As pointed out by [7], fast route computation is crucial given the scale of today's data centers.

session), and we only focus on flows that have significant volumes (which can be detected by, for example, [10]).

As concentrating these mega flows may severely congest some link(s), existing approaches such as ECMP (Equal-Cost Multi-Path forwarding [11]) apply only local decisions (at each switch) to balance the traffic loads. Such traffic-oblivious mechanisms may work well for wired networks, but they face difficulties in a hybrid DCN due to the existence of directional wireless links. In particular, while there are many possible outgoing wireless links from a node equipped with a directional antenna, one will be fixed for a certain time period once a routing decision is made, and hence the later routing requests cannot be oblivious to this earlier decision. For example, as shown in Fig. 2, once the two links $(7, 11)$ and $(12, 13)$ are set up, it may not be possible to set up link $(7, 12)$ anymore. To tackle this problem, we aim to mitigate the occupation level of each link by employing a joint routing and (antenna) scheduling scheme from a global perspective. More specifically, we identify a low-congestion routing path for each request. If several routing paths share a common 60GHz directional radio, we need to schedule the orientations of this radio and assign proper working times for them, such that the maximum congestion on all links is minimized.

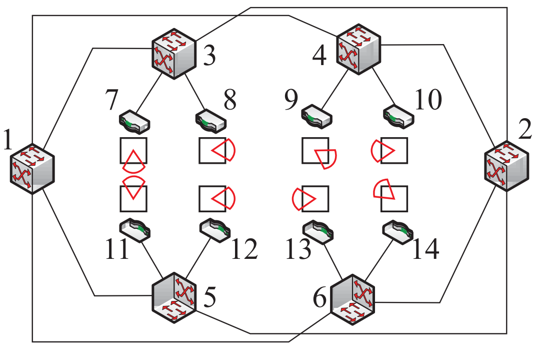


Fig. 2. An example for the joint routing and scheduling problem in a hybrid DCN. A wireless link exists between two antennas facing each other.

We formally define the joint routing and scheduling problem in hybrid DCNs by ***Definition** 1-3*:

***Definition** 1:* (*Feasible Schedule*) Given a hybrid DCN $G = (V, E)$, a feasible schedule τ for the wireless links in G is a set $\{\tau(e)|e \in E_W\}$ such that $\forall v \in V_R : \sum_{e\in\partial(v)} \tau_L(e) \leq 1$. The value $\tau(e)$ is called the *fractional time* scheduled by τ on $e, \forall e \in E_W$.
This means that, while we can steer a directional antenna to select different wireless links at different points in time, the time fractions allocated to different links should not overlap.

***Definition** 2:* (*Link Congestion*) Given a hybrid DCN $G = (V, E)$, a feasible schedule τ for E_W and a set $\ell = \{\ell(e)|e \in E\}$ of traffic loads on the links in E. The congestion on any $e \in E$ with respect to $\langle \ell, \tau\rangle$ is defined as:

$$\mathcal{C}^{\tau}_{\ell}(e) = \begin{cases} \ell(e)/c(e) & e \in E_D \\ \ell(e)/[c(e)\cdot\tau(e)] & e \in E_W \end{cases}$$

As $c(e)$ (resp. $c(e)\cdot\tau(e)$) is the effective capacity of a wired (resp. wireless) link given τ, reducing the congestion of links in G will leads to fewer "bottlenecks" in the network. This has the potential to improve throughput while reducing delay.

***Definition** 3:* (JRSH: *Joint Routing and Scheduling for Hybrid DCN*) Given a hybrid DCN G and a set of routing requests R, the JRSH problem seeks a path pt^*_i for each $r_i (1 \leq i \leq K)$ and a feasible schedule τ^* for the wireless links in G such that $Z_{opt} = \max_{e\in E} \mathcal{C}^{\tau^*}_{\ell^*}(e)$ is minimized, where $\forall e \in E : \ell^*(e) = \sum_{i:e\in pt^*_i} b_i$.

In short, the objective of JRSH is to minimize the maximum link congestion in the whole DCN. This, to some extent, represents the overall throughput performance of the DCN.

We explain the above definitions by a few simple examples based on Fig. 2. Suppose that the capacity of each link (wired and wireless) is 1 and we have three routing requests $r_1 = (7, 8, 0.5)$, $r_2 = (13, 14, 0.5)$ and $r_3 = (8, 13, 0.5)$. If we only rely on wired DCN, then the best solution is to route the three requests by the paths $pt_1 = 7 \to 3 \to 8$, $pt_2 = 13 \to 6 \to 14$ and $pt_3 = 8 \to 3 \to 2 \to 6 \to 13$, resulting in the maximum congestion of 1 on link $(8, 3)$ and $(13, 6)$. Using a wireless link $8 \dashrightarrow 13$ to replace pt_3, we end up with a better solution with the the maximum congestion reduced to 0.5.

According to ***Definition** 3*, we can formulate the JRSH problem as follows:

$$\begin{array}{rclll}
\text{Minimize} & & Z & & \text{[JRSH-1]} \\
\sum_{e:e\in out(s_i)} f^i_e & = & \sum_{e:e\in in(s_i)} f^i_e + 1 & \forall i & (1) \\
\sum_{e:e\in in(w)} f^i_e & = & \sum_{e:e\in out(w)} f^i_e & \forall i, \forall w \notin \{s_i, d_i\} & (2) \\
\sum_i f^i_e \cdot [b_i/c(e)] & \leq & Z & \forall e \in E_D & (3) \\
\sum_i f^i_e/c(e) & \leq & t_e \cdot Z & \forall e \in E_W & (4) \\
\sum_{e:e\in\partial(v)} t_e & \leq & 1; & \forall v \in V_R & (5) \\
f^i_e & \in & \{0, 1\} & \forall i, \forall e \in E & (6) \\
0 \leq t_e & \leq & 1 & \forall e \in E_W &
\end{array}$$

In the above formulation, the integer variable $f^i_e : \forall e \in E \wedge 1 \leq i \leq K$ indicates whether link e is used for routing r_i, and the variable $t_e : \forall e \in E_W$ corresponds to the fractional time scheduled on the wireless link $e \in E_W$. Constraints (1), (2) and (6) ensure that a single path is used for routing each request[3], where $out(v)$ and $in(v)$ denote the sets of outgoing links from v and incoming links to v, respectively. Constraint (3) and (4) bound the congestions on the wired and wireless links respectively, and constraint (5) guarantees that $\{t_e|e \in E_W\}$ is a feasible schedule.

The hardness of JRSH is immediate:

***Theorem** 1:* The JRSH problem is NP-hard.

For the convenience of further descriptions, we denote by b_{max} and b_{min} the maximum and the minimum traffic demands among all the requests, respectively, and denote by

[3]Multi-path routing can cause the problem of packet-reordering, hence is not considered in this paper.

c_{max} and c_{min} the maximum and the minimum link capacity among all links in E, respectively.

III. An Algorithm for Batched Request Arrival

JRSH-1 apparently belongs to the class of Mixed-Integer Nonlinear Programming (MINP) problems [12], which are usually very difficult to solve. In this section, we first convert JRSH-1 to an equivalent but more tractable form, then we propose RUSH-Batch as a fast approximation to tackle the problem, assuming that the requests in R arrive in a batch. To transform JRSH-1, we bring forward the following theorem.

***Theorem** 2:* Given $\ell = \{\ell(e)|e \in E\}$ as an arbitrary set of traffic loads on E and Λ as the set of all feasible schedules on the wireless links in E_W, we have:

$$\min_{\tau\in\Lambda}\max_{e\in E_W} \mathcal{C}_\ell^\tau(e) = \max_{v\in V_R}\sum\nolimits_{e\in\partial(v)}\frac{\ell(e)}{c(e)} \qquad (7)$$

The proof of ***Theorem** 2* actually reveals that, once the traffic loads on the wireless links in E_W are determined, we can find an optimal feasible schedule such that the maximum congestion on any $e \in E_W$ is minimized. Hence, if we consider the value of $\sum_{e\in\partial(v)}\frac{\ell(e)}{c(e)}$ as the "node congestion" of any wireless node v with respect to ℓ, then bounding the congestion of wireless nodes is equivalent to bounding the congestion of wireless links. Based on this observation, we convert JRSH-1 into the following equivalent Integer Linear Programming (ILP) problem:

$$\begin{array}{rcll}
\text{Minimize} & & Z & \text{[JRSH-2]} \\
\displaystyle\sum_{e:e\in out(s_i)} f_e^i & = & \displaystyle\sum_{e:e\in in(s_i)} f_e^i + 1 & \forall i \quad (8) \\
\displaystyle\sum_{e:e\in in(w)} f_e^i & = & \displaystyle\sum_{e:e\in out(w)} f_e^i & \forall i, \forall w \notin \{s_i, d_i\} \quad (9) \\
\sum\nolimits_i f_e^i\cdot[b_i/c(e)] & \le & Z & \forall e \in E_D \quad (10) \\
\displaystyle\sum_i\sum_{e:u\in e} f_e^i\cdot\frac{b_i}{c(e)} & \le & Z; & \forall u \in V_R \quad (11) \\
Z & \ge & b_{max}/c_{max} & (12) \\
f_e^i & \in & \{0,1\}; & \forall i, \forall e \in E \quad (13)
\end{array}$$

We can see that constraint (11) is set up in JRSH-2 to replace constraint (4) and constraint (5) in JRSH-1, and we completely removed the variables $t_e : e \in E_W$. According to the proof of ***Theorem** 2*, we can get the optimal routing and scheduling solution to the JRSH problem once JRSH-1 is solved. However, directly solving JRSH-2 can be time-prohibitive, since the JRSH problem is NP-hard. Hence, we seek to find an approximate solution to JRSH-2 by using a randomized rounding method, shown in **Algorithm 1**. To do this, we need a redundant constraint (12) in JRSH-2, which will be useful for bounding the integrality gap of the LP-relaxation of JRSH-2.

We relax JRSH-2 by replacing constraint (13) with $f_e^i \in [0,1]$; this LP-relaxation can be solved by a standard LP-solver. Its (optimal) fractional solution is taken as input to our RUSH-Batch algorithm, and we randomly choose routing-paths according to the fractional solution (lines 1-8). More specifically, we construct a set Γ_i of candidate paths for each request r_i, and each path $pt \in \Gamma_i$ is associated with a path probability $\xi(pt)$ (lines 2-7). According to constraint (8) of JRSH-2, we have $\sum_{pt\in\Gamma_i}\xi(pt) = 1$, and hence we can pick one path from $pt \in \Gamma_i$ according to the path probabilities (line 8). After choosing all the paths, we calculate the node congestions of all wireless nodes (lines 9-12) and then find a feasible antenna schedule for the wireless links using the method provided in the proof of ***Theorem** 2* (lines 13-14).

Algorithm 1: RUSH-Batch

```
Input: Request set $R$, the solution to the LP-relaxation
       of JRSH-2 $\{f_e^i | e \in E, 1 \le i \le K\}$
Output: Routing paths $\{pt_i\}_{i=1,\cdots,K}$ as well as a
        feasible schedule of wireless links $\{t_e\}_{e\in E_W}$
1  for $i = 1$ to $K$ do
2      $\Gamma_i \leftarrow \emptyset$
3      while $\exists e : f_e^i \neq 0$ do
4          Find $h \in E$ s.t. $f_h^i = \min\{f_e^i | e \in E \wedge f_e^i > 0\}$
5          Find a single path $pt$ from $s_i$ to $d_i$ that contains
           $h$ from the edge set $\{e \in E | f_e^i > 0\}$
6          $\xi(pt) \leftarrow f_h^i$;  $\Gamma_i \leftarrow \Gamma_i \bigcup \{pt\}$
7          forall the $e \in pt$ do $f_e^i \leftarrow f_e^i - \xi(pt)$
8      Pick a path $pt_i$ from $\Gamma_i$ such that the probability of
       selecting any $pt \in \Gamma_i$ is $\xi(pt)$
9  forall the $u \in V_R$ do $l(u) \leftarrow 0$
10 for $i = 1$ to $K$ do
11     Let $W$ be the set of all wireless links in $pt_i$
12     $\forall u \in e \in W : l(u) \leftarrow l(u) + b_i/c(e)$
13 $l_{max} \leftarrow \max\{l(u) | u \in V_R\}$
14 The fractional time scheduled on any wireless link $e$ is
   set to $t_e = \sum_{i:e\in pt_i} b_i/(c(e)\cdot l_{max})$
```

Lines 1-8 of **Algorithm 1** can be implemented in $\mathcal{O}(K|E|(|V| + |E|))$ time. Lines 9-14 of **Algorithm 1** can be implemented in $\mathcal{O}(K|E| + |V|)$ time. Hence the time complexity of **Algorithm 1** is $\mathcal{O}(K|E|^2)$. To bound the quality of the solution, we define the *Competitive Ratio* (CR) of any algorithm $\mathcal{A}$ for the JRSH problem as the ratio of the maximum link congestion resulting from $\mathcal{A}$ to Z_{opt}, and prove the CR of **Algorithm 1** by ***Lemma** 1* and ***Theorem** 3*:

***Lemma** 1:* Let $X_1, X_2, ..., X_k$ are independent random variables such that $\forall 1 \le i \le k$: $\text{Prob}\{X_i = 1\} = p_i$ and $\text{Prob}\{X_i = 0\} = 1 - p_i$. Let $Y = \sum_{1\le i\le k} a_i X_i$ where $a_i : 1 \le i \le k$ are non-negative real numbers. Let m be any positive number which satisfies $m \ge \max\{a_i | 1 \le i \le k\}$. Suppose $\mathbb{E}(Y) \le S$, then for any $\delta \ge 0$ we have: $\text{Prob}\{Y \ge (1+\delta)S\} \le \left[\mathbf{e}^\delta(1+\delta)^{-1-\delta}\right]^{\frac{S}{m}}$

***Theorem** 3:* The competitive ratio of **Algorithm 1** is $1 + \gamma\ln n$ with high probability, where $\gamma = \max\{\frac{3c_{max}}{c_{min}}, \mathbf{e}^2\}$, where $\mathbf{e}$ is the base of natural logarithms.

IV. An Online Competitive Algorithm

The RUSH-Batch algorithm we have proposed in Sec. III assumes that the routing requests arrive in a batch. However, it is also possible that the requests arrive in a sequential manner. If we start to make the routing/scheduling only after all the requests have arrived, the earlier requests will suffer a high delay. Therefore, we provide an online algorithm, RUSH-Online, to solve JRSH and also to handle requests in a timely manner. We provide the pesudocode of RUSH-Online in **Algorithm 2**, and we also explain the basic principles behind the algorithm design.

The idea of RUSH-Online is to route the requests according to a special weight assigning method and a global parameter θ, which is an estimation of the optimal solution Z_{opt}. We maintain a global variable P as the set of paths for already routed flows. Initially, P is an empty set and θ is set to a constant a (line 1, and we will explain later how we choose a). We also maintain a global variable g_e for each $e \in E_D$ and a global variable t_v for each $v \in V_R$; they are the current congestion values (normalized by θ) of e and v, respectively. We reset both g_e and t_v to 0 if either the request is the first one or the estimation of Z_{opt} is updated (lines 5-6).

The execution of RUSH-Online may consist of several rounds (line 3). In each round, the algorithm calculates the weights of the wired links and the wireless links according to two special exponential functions (lines 8-9), and then find a shorted path pt_i from s_i to d_i according to the weight assignment as the tentative routing path for r_i (line 10). After that, the algorithm calculates the (tentative) maximum new congestion values of each wired link and of each wireless node in pt_i (lines 11-12). If either value exceeds a predefined threshold (line 13), the algorithm judges that the tentative path pt_i may bring too much congestion and cannot be output as a result. This leads to a doubling of the estimation of Z_{opt} (i.e., doubles θ), and the algorithm starts another round to process r_i. Otherwise if pt_i passes the test in line 13, we accept pt_i as a valid path for r_i, re-schedule the wireless links in current known paths (line 17), and revises the congestion values accordingly (lines 18-22).

We obtain the competitive ratio of **Algorithm 2** using a method of amortized analysis. More specifically, we define a potential function:

$$\Pi(i) = \sum_{u\in V_R} \lambda^{t_u(i)}(2 - t^*_u(i)) + \sum_{e\in E_D} \lambda^{g_e(i)}(2 - g^*_e(i))$$

where $t_u(i), \forall u \in V_R$ and $g_e(i), \forall e \in E_D$ are the current (normalized) congestion values if pt_i is selected, whereas $t^*_u(i)$ and $g^*_e(i)$ are the corresponding congestion values in the optimal solution. We first prove in ***Lemma 2*** that Π is actually a decreasing function when $\lambda = \frac{4}{3}$. Moreover, we use ***Lemma 2*** to prove that **Algorithm 2** can process all requests without changing θ, as shown by ***Lemma 3***.

Lemma 2: If $Z_{opt} \le a$ and θ does not change, then $\Pi(i)$ is a decreasing function.

Lemma 3: If $Z_{opt} \le a$, then **Algorithm 2** can process all the requests without doubling θ.

Algorithm 2: RUSH-Online

```
Input: Request $r_i$, and global variables $P$, $\theta$, $\{g_e\}$, $\{t_v\}$
Output: Routing path $pt_i$ and a feasible schedule $\tau_P$ for
        all paths in $P$
1  if $i = 1$ then $P \leftarrow \emptyset$;  $\theta \leftarrow a$
2  $changed \leftarrow$ false;  $\lambda \leftarrow \frac{4}{3}$
3  repeat
4      if $i = 1 \lor changed$ then
5          forall the $e \in E_D$ do $g_e \leftarrow 0$
6          forall the $v \in V_R$ do $t_v \leftarrow 0$
7      forall the $e \in E$ do $\delta_e \leftarrow b_i/[c(e) \cdot \theta]$
8      forall the $e \in E_D$ do $w(e) \leftarrow \lambda^{g_e+\delta_e} - \lambda^{g_e}$
9      forall the $e \in E_W$ do
           $w(e) \leftarrow \lambda^{t_u+\delta_e} + \lambda^{t_v+\delta_e} - \lambda^{t_u} - \lambda^{t_v}$, where $u$ and $v$
           are $e$'s two endpoints
10     Find a shortest path $pt_i$ for routing $r_i$ using $w$
11     $z_1 \leftarrow \max\{g_e + \delta_e | e \in pt_i \bigcap E_D\}$
12     $z_2 \leftarrow \max\{t_u + \delta_e | u \in e \land e \in pt_i \bigcap E_W\}$
13     if $\max\{z_1, z_2\} > 3\log_\lambda n$ then
14         $\theta \leftarrow 2 \cdot \theta$,  $changed \leftarrow$ true
15     else $changed \leftarrow$ false
16 until $\neg changed$;
17 $P \leftarrow P \bigcup \{pt_i\}$;  Find a feasible schedule $\tau_P$ for the
   wireless links in $\bigcup_{pt\in P}(pt \cap E_W)$ using the same
   method with lines 9-14 of Algorithm 1
18 foreach $e = (u, v) \in pt_i$ do
19     if $e \in E_D$ then
20         $g_e \leftarrow g_e + \delta_e$
21     else
22         $t_u \leftarrow t_u + \delta_e$;  $t_v \leftarrow t_v + \delta_e$
```

Now the problems left are how to choose a and how to get the CR of **Algorithm 2**. Since Z_{opt} is unknown, we cannot guarantee that an initially chosen a is larger than Z_{opt}, hence **Algorithm 2** may run several rounds to get the right estimation of Z_{opt}. Fortunately, ***Lemma 3*** actually guarantees that **Algorithm 2** terminates in finite time: at most $\mathcal{O}(\log(Z_{opt}))$ rounds no matter what initial value is chosen for a. Also, we can use ***Lemma 3*** to choose a suitable a to bound the CR of **Algorithm 2** no matter how many times θ doubles itself, which results in ***Theorem 4***.

Theorem 4: If $a \in (0, \frac{b_1}{c_{max}}]$, then the competitive ratio of **Algorithm 2** is $12\log_\lambda n$ where $\lambda = \frac{4}{3}$.

Note that the interval $(0, \frac{b_1}{c_{max}}]$ can be determined once the first request r_1 shows up, hence **Algorithm 2** works in a pure online fashion, i.e., processing r_i without the knowledge of any future requests. According to ***Lemma 3***, there are at most $\mathcal{O}(\log(Z_{opt}))$ rounds in **Algorithm 2**. In each round, the time used for processing request r_i is at most $\mathcal{O}(|E| + |V|\log|V|)$. On the other hand, lines 17-22 can be implemented in $\mathcal{O}(|E| + |V|)$ time. Hence the total time complexity of **Algorithm 2** for processing all K requests is at most $\mathcal{O}(K(|E| + |V|\log|V|)\log(Z_{opt})) = \mathcal{O}(Kn^2\log(\frac{Kb_{max}}{c_{min}}))$.

V. Evaluating RUSH

We report our evaluation of RUSH in this section. We first introduce the configurations of the simulator. Then we compare the performance of our own algorithms and choose one to conduct the remaining simulations. Finally, we choose three typical traffic patterns to evaluate our RUSH-Online algorithm in middle scale hybrid DCNs.

A. *Simulation Settings*

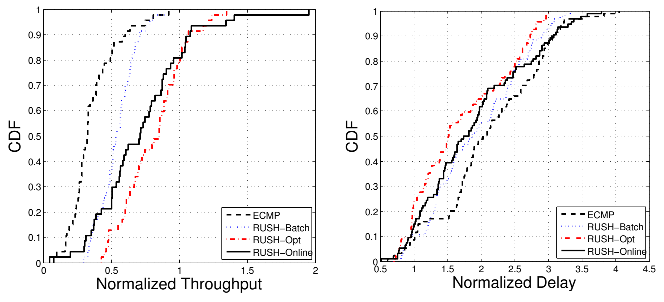
We implement RUSH in *ns*-3 and use CPLEX [13] as the ILP/LP solver. In the simulations, we create DCN topologies based on [14] (similar to that in Fig. 1). For the wireless models, simulations and the layout of racks, we mimic those used in [4], and the rates of wireless links are selected based on the conservative SINR model adopted in [4]. The rates of the wired links under a ToR are set to 1Gbps, and links at higher stages are configured to obtain a suitable oversubscription ratio. Similar to [1] and [15], we generate two types of routing requests: i) Stride-i flows: where a server with id x sends data to the destination server with id $(x+i) \mod n$, and ii) Random flows: where each server randomly selects another server as the destination. In either case, we generate flows whose start time follows a Poisson arrival process with a given mean.

We study the performance of five algorithms in our simulations, namely: RUSH-OPT, RUSH-Batch, RUSH-Online, Greedy, and ECMP. RUSH-OPT is the optimal solution of the JRSH problem, i.e., the results of solving JRSH-2 using an ILP-solver. Due to the high time complexity of JRSH-2, we only run RUSH-OPT for small scale hybrid DCNs and use it as a benchmark. RUSH-Batch and RUSH-Online are our algorithms described in Section III and Section IV, respectively. The Greedy algorithm corresponds to the heuristic provided in [4], which offloads the congested wired links using the available wireless links in a greedy manner. Finally, ECMP is the algorithm widely adopted by commercial DCNs. Among all these algorithms, only ECMP runs on purely wired DCNs, and we use it as a benchmark to show how much performance gain we can obtain by employing wireless links.

Although our algorithms all aim at minimizing link congestion, it is hard to measure link congestion in practice (in simulations or experiments). Therefore, we actually measure the throughput and the average per packet delay of the flows, and we normalize the two quantities against those of a non-oversubscribed (wired) DCN. As many flows are under evaluation, we plot their normalized throughput/delay in the form of empirical Cumulative Distribution Function (CDF).

B. *Simulation Results*

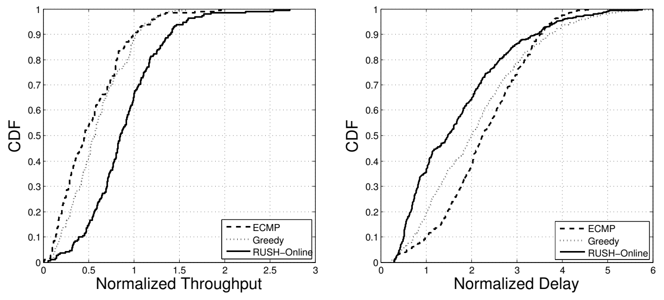
We first compare RUSH-OPT, RUSH-Batch, RUSH-Online and ECMP in small scale DCNs. Due to the high complexity of RUSH-OPT, we can only have DCNs with 28 switches (16 ToRs, 8 aggregations, and 4 cores) and 48 flows. The link rates are set to make the wired DCN 1:2 oversubscribed. The simulation results are shown in Fig. 3. It is not a surprise that all the three RUSH algorithms outperform the ECMP algorithm, proving the advantage of using wireless links to improve the DCN performance. While RUSH-OPT does appear to be better than both RUSH-Online and RUSH-Batch, the difference is too small to warrant its complexity. In the remaining simulations, we will use RUSH-Online as a representative to compare with other algorithms.



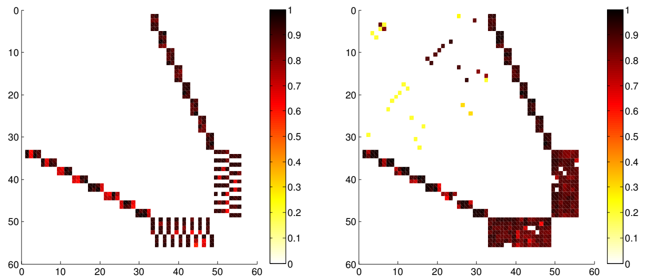
(a) CDF of throughput (b) CDF of delay

Fig. 3. Comparing the RUSH algorithms with ECMP.

Now we switch to middle scale DCNs with 56 switches (32 ToRs, 16 aggregations, and 8 cores) and 192 flows, and the link rates are still set to make the wired DCN 1:2 oversubscribed. Considering three flow patterns: Stride-6, Stride 12, and Random, we compare RUSH-Online with Greedy and ECMP under each pattern, and also investigate the traffic load of each link under both ECMP and RUSH-Online for Stride-6. We first show the comparison under Stride-6 in Fig. 4(a) and Fig. 4(b). Apparently, RUSH-online and Greedy outperform



(a) CDF of throughput (b) CDF of delay



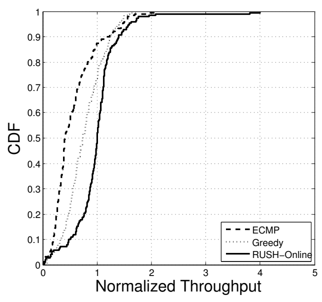
(c) Link load matrix of ECMP (d) Link load matrix of RUSH-Online

Fig. 4. Comparing RUSH-Online with Greedy and ECMP under Stride-6.

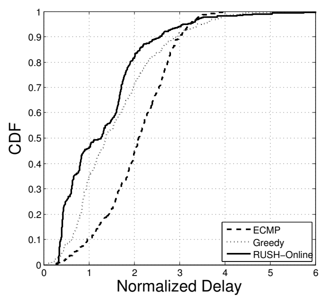
ECMP in terms of both throughput and delay, as they both take the advantage of wireless offloading. However, RUSH-Online surpasses Greedy significantly: about 35% of the flows routed by RUSH-Online achieve a normalized throughput of at least 1, whereas only less than 12% of flows routed by Greedy achieve the same throughput. This can be explained by the reason that the Greedy algorithm lacks of a flexible antenna scheduling scheme and selects fixed wireless links based on a local-optima-searching method for any given batch of flows, hence can not fully explore the power of wireless-offloading to balance the network traffic as RUSH-Online does. In fact, Greedy needs to let every flow run in order to identify which one(s) should be offloaded to wireless links. On the contrary, RUSH-Online determines the path for a flow before sending it into the network.

The wireless offloading achieved by RUSH-Online is very evidently shown by Fig. 4(c) and Fig. 4(d), where the color at i-th row and j-th column of a matrix indicates the traffic load on the link between a switch i and another switch j. As the switches are indexed such that the first 32 are ToRs and the last 8 are cores, the upper-left corner of a matrix represent the wireless links, whereas the rest portions represent the wired links. As ECMP only use wired links, bottlenecks are shown to be at the core switches. Instead, RUSH is able to spread the load on both wired and wireless links, leading to a lower congestion on links.

We further compare RUSH-Online with Greedy and ECMP under two other flow patterns: Stride-12 and Random; the results are shown in Fig. 5 and Fig. 6. In these cases, the superiority of RUSH-Online persists. As the queueing delay

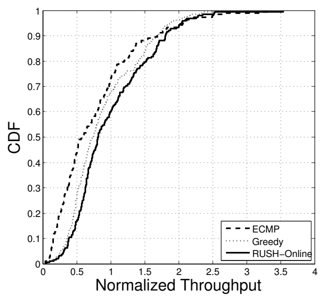


(a) CDF of throughput

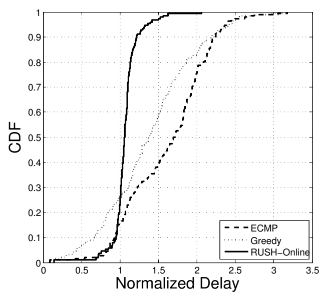


(b) CDF of delay

Fig. 5. Comparing RUSH-Online with Greedy and ECMP under Stride-12.



(a) CDF of throughput



(b) CDF of delay

Fig. 6. Comparing RUSH-Online with Greedy and ECMP under Random.

in a DCN appears to be minor and the transmission delay is the dominating factor, RUSH is able to improve both throughput and average (per packet) delay. However, as throughput is a cumulative quantity whereas delay is measured for every packet, the improvements in throughput often appear to be more prominent.

VI. Related Work

A huge body of work has been done for improving the performance of DCNs during the last couple of years; we only survey a few here for brevity. FatTree [1], Portland [16] and VL2 [2] have leveraged FatTree or Clos-type topologies to improve the routing efficiency in DCNs. Building upon a FatTree topology, Hedera [15] collects the flow information in DCNs to a central controller and optimize the flow routing using a simulated annealing method. BCube [3] and DCell [17] further allow servers to communicate directly and use highly connected network structures to improve routing performance. There are quite some drawback with all these purely wired DCN topologies (we refer to the comments in [4], [5] for details); a prominent one is their inflexibility in accommodating topology modifications or upgrades.

Recently, the hybrid DCNs with wireless links have started to attract attentions. The survey paper [18] gives a comprehensive analysis on the challenges of wireless DCNs, which greatly motivates potential research directions in this area. The work in [4], [5] has used both simulations and system implementations to demonstrate the feasibility of augmenting wired DCNs with 60GHz wireless links formed by highly directional antennas, but neither of them has provided joint routing and scheduling schemes for hybrid DCNs with provable performance bounds. In [6], the channel allocation problem in wireless DCNs has also been investigated and some novel algorithms are provided to efficiently increase the network throughput and reduce the job completion time.

The related disjoint-paths searching and congestion-aware routing problems were also studied in [19]–[22]. However, these proposals all work for static network graphs and can only be applied to traditional wired networks. Moreover, there are also a large number of proposals for routing and scheduling using directional antennas in wireless ad hoc networks, e.g., [23]–[25], but they focus on the specific features and optimization goals for wireless ad hoc networks (such as lifetime and coverage), which are essentially different from those in hybrid DCNs.

VII. Conclusion

In this paper, we have studied, from a system perspective, the network flow optimization problem in hybrid CDNs, where highly directional 60GHz wireless antennas are deployed for augmenting the wired infrastructure of traditional DCNs. We have proposed our RUSH framework for joint routing and scheduling directional wireless antennas in the hybrid DCNs, and present algorithms to deal with both the batch-arrival and the sequential-arrival of network flow demands. These algorithms all offer provable performance guarantees in terms of the network congestion. We have implemented the RUSH framework in *ns*-3, and by using the implementations, we have conducted extensive simulations in *ns*-3 to compare RUSH with existing proposals. The results have strongly demonstrated the superiority of our approach.

ACKNOWLEDGMENT

This work was supported in part by AcRF Tier 1 Grant RGC5/13 and the National Natural Science Foundation of China (No. 61472460).

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, “A Scalable, Commodity Data Center Network Architecture,” in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.
[2] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, “VL2: A Scalable and Flexible Data Center Network,” in *Proc. ACM SIGCOMM*, 2009, pp. 51–62.
[3] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, “BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers,” in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.
[4] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, “Augmenting Data Center Networks with Multi-Gigabit Wireless Links,” in *Proc. ACM SIGCOMM*, 2011, pp. 38–49.
[5] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Zhao, and H. Zheng, “Mirror Mirror on the Ceiling: Flexible Wirelesss Links for Data Centers,” in *Proc. ACM SIGCOMM*, 2012.
[6] Y. Cui, H. Wang, and X. Cheng, “Channel Allocation in Wireless Data Center Networks,” in *Proc. IEEE INFOCOM*, 2011, pp. 1395–1403.
[7] T. Benson, A. Akella, and D. Maltz, “Network Traffic Characeteristic of Data Centers in the Wild,” in *Proc. ACM IMC*, 2010, pp. 267–280.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks,” *ACM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[9] X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste, “Dirc: increasing indoor wireless capacity using directional antennas,” in *Proc. ACM SIGCOMM*, 2009, pp. 171–182.
[10] Z. Hu and J. Luo, “Cracking Network Monitoring in DCNs with SDN,” in *Proc. IEEE INFOCOM*, 2015.
[11] C. Hopps, “Analysis of an Equal-Cost Multi-Path Algorithm,” RFC 2992, 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2992.txt
[12] J. Lee and S. Leyffer, *Mixed Integer Nonlinear Programming*. Berlin: Springer Verlag, 2012.
[13] “IBM ILOG CPLEX Optimizer.” [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimizer/
[14] C. D. C. Infrastructure, “2.5 Design Guide.” [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf
[15] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: dynamic flow scheduling for data center networks,” in *Proc. USENIX NSDI*, 2010, pp. 19–19.
[16] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, “Portland: a scalable fault-tolerant layer 2 data center network fabric,” in *Proc. ACM SIGCOMM*, 2009, pp. 39–50.
[17] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, “Dcell: a scalable and fault-tolerant network structure for data centers,” in *Proc. ACM SIGCOMM*, 2008, pp. 75–86.
[18] Y. Cui, H. Wang, X. Cheng, and B. Chen, “Wireless data center networking,” *Wireless Communications, IEEE*, vol. 18, no. 6, pp. 46–53, December 2011.
[19] P. Raghavan and C. D. Thompson, “Provably good routing in graphs: regular arrays,” in *Proc. ACM STOC*, 1985, pp. 79–87.
[20] A. Baveja and A. Srinivasan, “Approximation algorithms for disjoint paths and related routing and packing problems,” *Math. Oper. Res.*, vol. 25, no. 2, pp. 255–280, 2000.
[21] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, “Online load balancing with applications to machine scheduling and virtual circuit routing,” in *Proc. ACM STOC*, 1993, pp. 623–631.
[22] C. Chekuri, S. Khanna, and F. B. Shepherd, “Edge-disjoint paths in planar graphs with constant congestion,” in *Proc. ACM STOC*, 2006, pp. 757–766.
[23] L. Bao and J. Garcia-Luna-Aceves, “Transmission scheduling in ad hoc networks with directional antennas,” in *Proc. ACM MobiCom*, 2002, pp. 48–58.
[24] A. Orda and B.-A. Yassour, “Maximum-lifetime routing algorithms for networks with omnidirectional and directional antennas,” in *Proc. ACM MobiHoc*, 2005, pp. 426–437.
[25] K. Han, L. Xiang, J. Luo, and Y. Liu, “Minimum-energy Connected Coverage in Wireless Sensor Networks with Omni-directional and Directional Features,” in *Proc. of ACM MobiHoc*, 2012, pp. 85–94.
[26] R. Motwani and P. Raghavan, *Randomized algorithms*. New York: Cambridge University Press, 1995.

APPENDIX

*Proof of **Theorem** 1:* The NP-hardness of the JRSH problem can be easily proved by a reduction from the the Edge-Disjoint Paths with Congestion problem, which is NP-hard even on planar graphs [22]. Hence we omit the proof details due to page limits. □

*Proof of **Theorem** 2:* Let the righthand of equation (7) be B. Firstly we'll prove that we can find a feasible schedule $\tau_1 \in \Lambda$ such that $\max_{e\in E_W} \mathcal{C}_\ell^{\tau_1}(e) \le B$, which leads to $\min_{\tau\in\Lambda}\max_{e\in E_W}\mathcal{C}_\ell^{\tau}(e) \le B$. The feasible schedule τ_1 can be constructed as follows. For any link $e \in E_W$, the fractional time scheduled by τ_1 on e is set to $\tau_1(e) = \ell(e)/(c(e)\cdot B)$. Clearly, $\mathcal{C}_\ell^{\tau_1}(e) \le B$. Meanwhile, for any $v \in V_R$, we have:

$$\sum\nolimits_{e\in\partial(v)} \tau_1(e) = (1/B)\cdot \sum\nolimits_{e\in\partial(v)} [\ell(e)/c(e)] \le 1,$$

which indicates that τ_1 is a feasible schedule for E_W and satisfies $\max_{e\in E_W}\mathcal{C}_\ell^{\tau_1}(e) \le B$.

Let v^* be the specific node in V_R such that the maximum value (i.e., B) of the right side of equation (7) is achieved. Now we'll prove $\min_{\tau\in\Lambda}\max_{e\in E_W}\mathcal{C}_\ell^{\tau}(e) \ge B$. To prove this, it is sufficient to prove $I(\tau) = \max_{e\in\partial(v^*)}\mathcal{C}_\ell^{\tau}(e) \ge B$ $(\forall\tau\in\Lambda)$. This can be proved by observing that, for any feasible schedule $\tau\in\Lambda$, we have $\sum_{e\in\partial(v^*)}\tau(e) \le 1$ and $\forall e \in \partial(v^*) : \mathcal{C}_\ell^{\tau}(e) = \ell(e)/[\tau(e)\cdot c(e)] \le I(\tau)$, hence $B/I(\tau) = \sum_{e\in\partial(v^*)}\ell(e)/[I(\tau)\cdot c(e)] \le 1$. □

*Proof of **Theorem** 3:* For each $e \in E$ and each $i(1 \le i \le K)$, define a random variable Y_e^i such that: if the link e is selected for routing r_i, then $Y_e^i = 1$, otherwise $Y_e^i = 0$. Hence we have $\mathbb{E}(Y_e^i) = \sum_{e:e\in p\wedge p\in\Gamma_i}\xi(p) = f_e^i$. For any link $e \in E_D$, let $X_e = \sum_i \frac{b_i}{c(e)}\cdot Y_e^i$ be the congestion on e. Let Z^* be the optimal solution to the LP-relaxation of JRSH-2. Clearly, $\mathbb{E}(X_e) = \sum_i f_e^i\cdot\frac{b_i}{c(e)} \le Z^*$.

Let $\alpha = \gamma\cdot\ln n$ and $m = b_{max}/c_{min}$. Using ***Lemma** 1* and $Z^* \le Z_{opt}$, we have:

$$\begin{aligned}\mathrm{Prob}\{X_e > (1+\alpha)\cdot Z_{opt}\} &\le \mathrm{Prob}\{X_e > (1+\alpha)\cdot Z^*\}\\ &\le \left\{\mathbf{e}^{\alpha}(1+\alpha)^{-1-\alpha}\right\}^{\frac{Z^*}{m}} \le \left\{(\alpha/\mathbf{e})^{-\alpha}\right\}^{\frac{Z^*}{m}}\end{aligned}$$

On the other hand, since $\gamma = \max\{3c_{max}/c_{min}, \mathbf{e}^2\}$, we have $\ln\left(\frac{\alpha}{\mathbf{e}}\right)^{\alpha} = \gamma\ln n(\ln\gamma + \ln\ln n - 1) \ge \frac{3c_{max}}{c_{min}}\cdot\ln n$, hence $(\alpha/\mathbf{e})^{\alpha} \ge n^{\frac{3c_{max}}{c_{min}}}$. Combining all these inequalities with $Z^* \ge b_{max}/c_{max}$, we can get $\mathrm{Prob}\{X_e > (1+\alpha)\cdot Z_{opt}\} \le n^{-\frac{3c_{max}}{c_{min}}\cdot\frac{Z^*}{m}} \le n^{-\frac{3c_{max}}{c_{min}}\cdot\frac{b_{max}}{c_{max}}\cdot\frac{c_{min}}{b_{max}}} = n^{-3}$. Finally, according to the Boole's inequality we have $\mathrm{Prob}\{\exists e \in E_D : X_e > (1+\alpha)\cdot Z_{opt}\} \le |E_D|\cdot\mathrm{Prob}\{X_e > (1+\alpha)\cdot Z_{opt}\} \le 1/n$, which means that the congestion on any wired link is no more than $(1+\gamma \ln n)Z_{opt}$ with high probability. Similarly, we can prove that the congestion on any wireless node is no more than $(1+\gamma \ln n)Z_{opt}$ with high probability. According to ***Theorem** 2*, ***Theorem** 3* follows. □

*Proof of **Lemma** 1:* The lemma is actually a generalization of the Chernoff bound [26]. Let $t=\ln(1+\delta)$. For any variable x, let $\overline{x}=x/m$. Using Markov's inequality, we have

$$\begin{aligned}
&\text{Prob}\{Y \geq (1+\delta)S\} = \text{Prob}\{\overline{Y} \geq (1+\delta)\overline{S}\}\\
= \;& \text{Prob}\{\mathbf{e}^{t\overline{Y}} \geq \mathbf{e}^{t(1+\delta)\overline{S}}\}\\
\leq \;& \mathbb{E}(\mathbf{e}^{t\overline{Y}})/\mathbf{e}^{t(1+\delta)\overline{S}} = \mathbb{E}(\prod_{1\leq i\leq k} \mathbf{e}^{t\overline{a_i}Xi})/\mathbf{e}^{t(1+\delta)\overline{S}}\\
= \;& \frac{\prod_{1\leq i\leq k}\mathbb{E}(\mathbf{e}^{t\overline{a_i}Xi})}{\mathbf{e}^{t(1+\delta)\overline{S}}} = \frac{\prod_{1\leq i\leq k}[1+p_i(\mathbf{e}^{t\overline{a_i}}-1)]}{\mathbf{e}^{t(1+\delta)\overline{S}}}
\end{aligned}$$

Since $\overline{a_i}\in[0,1]$ $(\forall 1\leq i\leq k)$ and $t\geq 0$, we have: $1+p_i(\mathbf{e}^{t\overline{a_i}}-1)\leq 1+p_i\overline{a_i}(\mathbf{e}^t-1)\leq \mathbf{e}^{p_i\overline{a_i}(\mathbf{e}^t-1)}$, Therefore

$$\begin{aligned}
&\text{Prob}\{Y\geq(1+\delta)S\}\leq \frac{\mathbf{e}^{\sum_{1\leq i\leq k}[p_i\overline{a_i}(e^t-1)]}}{\mathbf{e}^{t(1+\delta)\overline{S}}}\\
&\leq \mathbf{e}^{\overline{S}(e^t-1)}\mathbf{e}^{-t(1+\delta)\overline{S}} = \left(\mathbf{e}^{\delta}(1+\delta)^{-1-\delta}\right)^{\frac{S}{m}}
\end{aligned}$$

□

*Proof of **Lemma** 2:* Let $\Pi_1(i)=\sum_{u\in V_R}\lambda^{t_u(i)}(2-t^*_u(i))$ and $\Pi_2(i)=\Pi(i)-\Pi_1(i)$. Let pt^*_i be the path for request r_i in the optimal solution. Let E^1_W be the set of wireless links in pt_{i+1} but not in pt^*_{i+1}. Let E^2_W be the set of wireless links in both pt_{i+1} and pt^*_{i+1}. Let E^3_W be the set of wireless links in pt^*_{i+1} but not in pt_{i+1}. We have:

$$\begin{aligned}
&\Pi_1(i+1)-\Pi_1(i)\\
= \;& \sum_{u:u\in e\in E^1_W}(\lambda^{t_u(i+1)}-\lambda^{t_u(i)})(2-t^*_u(i))\\
&+\sum_{u:u\in e\in E^2_W}[\lambda^{t_u(i+1)}(2-t^*_u(i)-\delta_e(i+1))-\\
&\lambda^{t_u(i)}(2-t^*_u(i))] - \sum_{u:u\in e\in E^3_W}\lambda^{t_u(i)}\cdot\delta_e(i+1)\\
= \;& \sum_{u:u\in e\in pt_{i+1}\bigcap E_W}(\lambda^{t_u(i+1)}-\lambda^{t_u(i)})(2-t^*_u(i))\\
&-\sum_{u:u\in e\in pt^*_{i+1}\bigcap E_W}\lambda^{t_u(i+1)}\cdot\delta_e(i+1)\\
\leq \;& \sum_{u:u\in e\in pt_{i+1}\bigcap E_W}2(\lambda^{t_u(i)+\delta_e(i+1)}-\lambda^{t_u(i)})\\
&-\sum_{u:u\in e\in pt^*_{i+1}\bigcap E_W}\lambda^{t_u(i)}\cdot\delta_e(i+1), \qquad (14)
\end{aligned}$$

where (14) holds because $t_u(i+1)\geq t_u(i)$ and $t^*_u(i)\geq 0(\forall u\in V_R)$. Similarly, we can get:

$$\begin{aligned}
&\Pi_2(i+1)-\Pi_2(i)\\
\leq \;& \sum_{e:e\in pt_{i+1}\bigcap E_D}2(\lambda^{g_e(i)+\delta_e(i+1)}-\lambda^{g_e(i)})\\
&-\sum_{e:e\in pt^*_{i+1}\bigcap E_D}\lambda^{g_e(i)}\cdot\delta_e(i+1),
\end{aligned}$$

hence

$$\begin{aligned}
&\Pi(i+1)-\Pi(i)\\
\leq \;& \sum_{e:e\in pt^*_{i+1}\bigcap E_D}2(\lambda^{g_e(i)+\delta_e(i+1)}-\lambda^{g_e(i)})\\
&+\sum_{u:u\in e\in pt^*_{i+1}\bigcap E_W}2(\lambda^{t_u(i)+\delta_e(i+1)}-\lambda^{t_u(i)})\\
&-\sum_{e:e\in pt^*_{i+1}\bigcap E_D}\lambda^{g_e(i)}\cdot\delta_e(i+1)\\
&-\sum_{u:u\in e\in pt^*_{i+1}\bigcap E_W}\lambda^{t_u(i)}\cdot\delta_e(i+1) \qquad (15)\\
= \;& \sum_{e:e\in pt^*_{i+1}\bigcap E_D}\lambda^{g_e(i)}[2(\lambda^{\delta_e(i+1)}-1)-\delta_e(i+1)]\\
&+\sum_{u:u\in e\in pt^*_{i+1}\bigcap E_W}\lambda^{t_u(i)}[2(\lambda^{\delta_e(i+1)}-1)-\delta_e(i+1)],
\end{aligned}$$

where (15) holds because that pt_{i+1} is a shortest path according to the weight assignment method in lines 8-9 of **Algorithm 2**. Now, since $\delta_e(i+1)=b_{i+1}/(c(e)\cdot\theta)\leq Z_{opt}/a\leq 1\quad(\forall e\in pt^*_{i+1})$, we know $2(\lambda^{\delta_e(i+1)}-1)-\delta_e(i+1)\leq 0$ $(\lambda=\frac{4}{3},\forall e\in pt^*_{i+1})$, which indicates that $\Pi(i)$ is a decreasing function. □

*Proof of **Lemma** 3:* Suppose θ remains to be a after i requests are successfully processed. Clearly this holds for $i=0$. Now suppose that we are going to process $r_i(i\geq 1)$. Since $t^*_u(i)\leq Z_{opt}/\theta=Z_{opt}/a\leq 1(\forall u\in V_R)$ and similarly $g^*_e(i)\leq 1(\forall e\in E_D)$, we have $\Pi(i)\geq\sum_{u\in V_R}\lambda^{t_u(i)}+\sum_{e\in E_D}\lambda^{g_e(i)}$. According to ***Lemma** 2*, if we choose pt_i for r_i then we have:

$$\begin{aligned}
t_u(i) &= \log_\lambda(\lambda^{t_u(i)})\leq\log_\lambda\Pi(i)\\
&\leq \log_\lambda\Pi(0)=\log_\lambda 2(|V_R|+|E_D|)\\
&\leq \log_\lambda 2n^2\leq 3\log_\lambda n \qquad (\forall u\in V_R)
\end{aligned}$$

Similarly, we can also get $g_e(i)\leq 3\log_\lambda n$ $(\forall e\in E_D)$. This means that **Algorithm 2** won't execute line 14, hence can successfully process r_i without changing θ. □

*Proof of **Theorem** 4:* Suppose that **Algorithm 2** doubles θ for $k(k\geq 0)$ times to process all requests. Hence we can divide the execution history of **Algorithm 2** into $k+1$ stages such that each stage has the same value of θ. Let the maximum congestion of any wired or wireless link accumulated in the ith stage be T^i. According to **Algorithm 2**, we have $T^i\leq 3\cdot(\log_\lambda n)\cdot(2^{i-1}a);(1\leq i\leq k+1)$.

Let T_{final} be the maximum congestion of any wired or wireless link when all the requests are processed. Note that $a\leq b_1/c_{max}\leq Z_{opt}$. So if $k=0$, then we have

$$T_{final}=T^1\leq 3\cdot(\log_\lambda n)\cdot a\leq 12\cdot(\log_\lambda n)\cdot Z_{opt}$$

Now assume $k\geq 1$. According to ***Lemma** 3*, this implies $Z_{opt}>2^{k-1}a$. Hence, we also have

$$\begin{aligned}
T_{final}/Z_{opt} &\leq \sum_{1\leq i\leq k+1}T^i/Z_{opt}\\
&\leq \frac{\sum_{1\leq i\leq k+1}3\cdot(\log_\lambda n)\cdot(2^{i-1}a)}{2^{k-1}\cdot a}\leq 12\log_\lambda n
\end{aligned}$$

□