Opportunistic Scheduling in Ferry-Based Networks

Shimin Guo, Majid Ghaderi, Aaditeshwar Seth, Srinivasan Keshav
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{sguo,mghaderi,a3seth,keshav}@cs.uwaterloo.ca

ABSTRACT

Ferry-based networks provide a means of bringing extremely low-cost Internet access to remote rural areas, where conventional access technologies, *e.g.*, dial-up, DSL and CDMA, are currently economically infeasible. We present an architecture for ferry-based networks and identify downlink scheduling as a difficult open problem. We argue that *opportunistic scheduling* policies that take ferry schedules into account are more suitable for such networks than the non-opportunistic policies such as round-robin. We then, for a simplified variant of the problem, propose an opportunistic scheduling algorithm aimed at minimizing the end-to-end delay incurred in the network. Through simulations, we show that the proposed opportunistic scheduling algorithm outperforms the non-opportunistic algorithms in reducing end-to-end delay while achieving a larger stability region. In typical scenarios we see a gain of up to 20% using our technique as compared to a naive approach.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Protocols*

General Terms

Design, Performance

Keywords

Ferry-Based Networks, Opportunistic Scheduling, Performance Analysis

1. INTRODUCTION

Ferry-based networks provide a means of bringing extremely low-cost Internet access to remote rural areas, where conventional access technologies such as DSL and CDMA are currently economically infeasible. A ferry-based network is depicted in Figure 1. A ferry-based network has four major components: rural kiosks, buses, Internet gateways, and a proxy server. Kiosks are where end users send and receive data. Buses serve as a *mechanical backhaul* [11], ferrying data between the kiosks and Internet gateways. Internet gateways are usually located in nearby towns or cities and forward data to the proxy, which is located somewhere on the Internet, through persistent connections such as dial-up or DSL. In the direction from the Internet to the kiosks, the proxy hides legacy servers from the fact that the users are actually disconnected. In the other direction, from kiosks to the Internet, the proxy communicates on behalf of users with legacy servers.

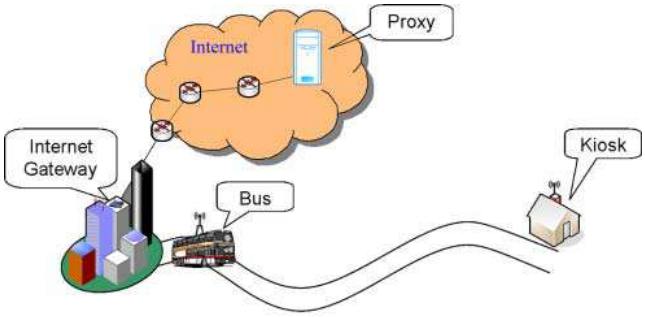

Figure 1: A ferry-based network.

In ferry-based networks, routing data between kiosks and the proxy is a challenging problem due to the following issues:

1. Time-dependent connection opportunities: Data is transmitted only when buses are at gateways or at kiosks.
2. Unreliable routes: A bus carrying data can fail due to mechanical (engine failure) or environmental (a closed road) problems.
3. Bottleneck links: The links between the gateways and the proxy have low capacity (compared to high-bandwidth WiFi connection between buses and gateways/kiosks) and can be easily congested, causing long delays at the proxy. Upgrading the bottleneck links, even if technically possible, results in increased operation costs that is undesirable for low-cost ferry-based networks. That is, over-provisioning is simply not an option. In order to reduce cost, it is imperative to fully utilize these links in an efficient manner.
4. Variation in schedules: Bus schedules exhibit slight variations that are not known either to the kiosks or the proxy.

5. Distinction between buses and routes: The same bus may run on different routes at different times of the day or on different days. The assignment may not be known to kiosks or the proxy.

6. Buses and kiosks have finite storage capacity.

In this paper, we focus on scheduling and ignore reliability, assuming that all paths are fully reliable. We also assume full knowledge of schedules. Therefore, sending data over a single path is sufficient to ensure reliable delivery of data. However, these paths have time-dependent delays. A path that is considered the shortest path at a specific point in time may become the longest path shortly thereafter. For example, data sent to a gateway just before a bus departure will arrive at a kiosk much earlier than data sent a few seconds after. Therefore, it is crucial to send data over the right path at the right time. Although the network is delay tolerant and the applications designed for such network, *e.g.*, email, are delay-tolerant in nature, it is still important to minimize end-to-end delay [3]. Decisions on when and where to send data are made by the scheduler residing at the proxy. Our goal is to achieve better delay performance by making more intelligent scheduling decisions at the proxy. The idea is to take bus schedule into consideration in making scheduling decisions.

The rest of the paper is organized as follows. Section 2 describes the system model and motivates the idea of opportunistic scheduling. In Section 3, we present our proposed opportunistic scheduling policies aimed at minimizing end-to-end delay in a ferry-base network. We show some simulation results in Section 4. Related work is summarized in Section 5. Finally, Section 6 concludes the paper and discusses some possible future research directions.

2. SYSTEM MODEL

A conceptual architecture of the ferry-based network is depicted in Figure 2. We now make a number of modelling simplifications. These serve to bring out the essential aspects of the problem. We consider a single proxy that connects the ferry-based network to the Internet through M gateways. There are N kiosks and there is at least one path from the proxy to each kiosk to ensure end-to-end connectivity. However, there may exist multiple paths from the proxy to a kiosk, potentially through multiple gateways. At the proxy, we assume there is an infinite buffer for each kiosk. Gateways, buses and kiosks all are also assumed to have infinite buffers.

A ferry-based network belongs to the family of delay-tolerant networks (DTN) [2]. Keeping with the DTN bundle protocol [10], the unit of data in the ferry-based network is called *bundle*. Let λ_i denote the bundle arrival rate destined to kiosk i at the proxy and μ_j denote the service rate of the link from the proxy to gateway j where these rates are long-term average rates. We assume that the proxy system is stable, that is

$$\rho = \frac{\sum_{i=1}^{N} \lambda_i}{\sum_{j=1}^{M} \mu_j} < 1,$$

where ρ denotes the total load of the proxy. Note that this

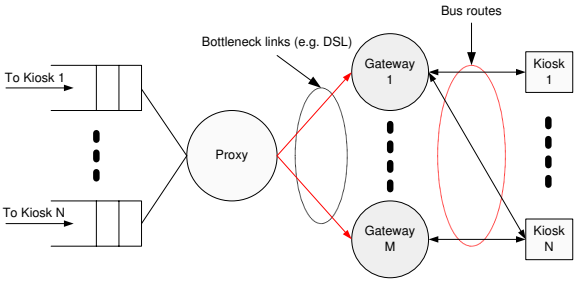

Figure 2: Each kiosk can be reached through multiple gateways.

is an overall stability criterion and poor scheduling decision may make one or more queues unstable. It is assumed that bus contact times with gateways are sufficiently long to upload/download all bundles between a bus and the corresponding gateway.

In the following two subsections, we describe the ferry-based network operation. We briefly describe both uplink and downlink data transmission with a focus on downlink operation.

2.1 Uplink Data Transmission

There is no bottleneck and hence, no competition for bandwidth among kiosks, in the uplink direction. Data transmission, therefore, is solely a routing problem. A modified version of the Dijksta's algorithm [3] that takes into consideration the time-dependent link delays can be used to find the shortest path from a kiosk to the best gateway to reach the proxy. However, this strategy may lead to unbalanced distribution of traffic among gateways.

A simple solution is to flood data to multiple gateways and apply a hand-shaking mechanism between gateways and the proxy to avoid multiple copies of the same data being transmitted to the proxy over the bottleneck links. That is, if a bundle has already been received by the proxy through a gateway, then no other gateway will send the same bundle to the proxy. This ensures that bundles are always sent on the shortest path yet the bottleneck resource is not wasted. In the more general case, when buses are subject to failures, this approach has the advantage of partially mitigating bus failures as well.

2.2 Downlink Data Transmission

The direction from the proxy to kiosks is much more complex because we cannot flood data on the bottleneck links. When the data for kiosk i arrives at the proxy, the proxy stores it in the queue associated with kiosk i. We call this queue i. Whenever a link from the proxy to a gateway becomes available, the proxy chooses from the set of queues, whose associated kiosks can be reached from that gateway by a bus, a bundle to transmit over the link. A scheduling policy dictates which queue to select for transmission.

Several candidates exist for the scheduling policy, such as round-robin, FCFS, or any flavour of weighted fair queueing. However, they all have one problem, that is, they are all *bus schedule-agnostic*, which means they do not take the

bus schedules into account when making scheduling decisions. This can lead to poor performance of the network with respect to end-to-end delay[1].

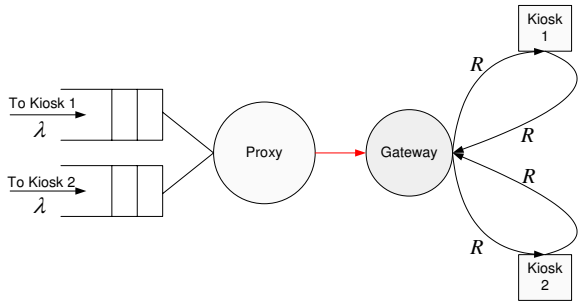

Figure 3: Downlink scheduling.

To illustrate this, consider the following elementary scenario. Suppose we have one gateway, two buses, and two kiosks, as illustrated in Figure 3. Two buses depart from the gateway, each serving a different kiosk. The system is completely symmetric with respect to the kiosks in that the two kiosks have the same arrival rate and similar bus routes. Transit time between the gateway and kiosks is R minutes in both directions. However, whenever bus 1 arrives at the gateway bus 2 is at kiosk 2 and vice versa. The link between the proxy and the gateway serves a bundle every $1/2\lambda$ minutes, *i.e.*, $\mu = 2\lambda$. In the following, we consider two deterministic bundle arrival processes. As we will see shortly, these arrival processes are two examples of extreme arrival processes for the network in Figure 3.

- *Process (i):* Bundles arrive at the proxy every $1/\lambda$ minutes for both kiosks.
- *Process (ii):* Bundles arrive in batches of size $2R\lambda$ every round-trip-time, *i.e.*, every $2R$ minutes. As soon as bus 1 leaves kiosk 1 (bus 2 leaves the gateway), a batch of bundles arrives at the proxy for kiosk 1 (for kiosk 2).

First, consider a round-robin or any variation of fair queueing scheduling in which the link capacity from the proxy to the gateway is equally shared between the two kiosks. Then, for Process (i), every time a bus arrives at the gateway it picks up all the bundles and hence, the corresponding queue at the proxy will be empty immediately after the bus leaves. However, for Process (ii), only $R\lambda$ bundles from the current batch are picked up (plus $R\lambda$ bundles from the previous batch) by bus 1. The remaining $R\lambda$ bundles wait at the gateway for the next arrival of the bus which will happen $2R$ minutes later. Therefore, for kiosk 1, $R\lambda$ bundles experience extra $2R$ minutes of delay at every round.

Now, consider a different scheduling policy which gives priority to the kiosk whose bus will arrive at the gateway sooner. We call this policy *bus-schedule-aware* policy. In this case, each kiosk has the bottleneck link for R minutes that takes the corresponding bus to arrive at the gateway from the kiosk. Therefore, by the time the bus arrives at the gateway all $2R\lambda$ bundles are already at the gateway (note that service rate is 2λ). It means that no bundle will experience extra delay under this scheduling policy regardless of the traffic arrival process, *i.e.*, Process (i) or Process (ii).

As mentioned before, this special example considers two extreme arrival processes. Let $N_f(i)$ denote the number of bundles arrive at the proxy for kiosk i when the corresponding bus is in its forward route, *i.e.*, from the gateway to the kiosk. Similarly, let $N_r(i)$ denote the number of bundles arrive at the proxy for kiosk i when the bus is in its reverse route, *i.e.*, from the kiosk to the gateway. In general, if $N_r(1) > N_f(1)$ and $N_r(2) < N_f(2)$, then the bus-schedule-aware policy achieves shorter end-to-end delay than round-robin like policies. It is straightforward to construct other scenarios using these arrival processes. For example, if one kiosk has arrival process (i) and the other has arrival process (ii), again the bus-schedule-aware policy achieves shorter end-to-end delay than round-robin policy. However, the bus-schedule-aware policy is never worse than round-robin. This simple example clearly shows that it is possible to reduce the end-to-end delay of one kiosk *without increasing the delay of other kiosks.*

We now present a more formal argument in favour of the bus-schedule-aware scheduling policy. The end-to-end delay, as we have define in this paper, consists of three parts: (1) queueing delay at the proxy denoted by D, (2) waiting time at the gateway for the bus to arrive denoted by W, and, (3) transit time on the bus denoted by R. Let T denote the end-to-end delay. Then

$$T = D + W + R\,.$$

The goal of scheduling is to minimize the expected end-to-end delay, that is to minimize $\mathbb{E}[T]$, where

$$\mathbb{E}[T] = \mathbb{E}[D + W + R] = \mathbb{E}[D] + \mathbb{E}[W] + \mathbb{E}[R]\ .$$

Since the proxy system is stable, $\mathbb{E}[D]$ is finite and constant. Using appropriate queueing models, $\mathbb{E}[D]$ can be computed. For the family of work-conserving policies we can model the proxy as a single server queue with arrival rate 2λ and service rate $\mu > 2\lambda$. For example if the arrival process is Poisson and bundles have fixed size, $\mathbb{E}[D]$ is the mean waiting time in an $M/D/1$ queue. Therefore,

$$\mathbb{E}[D] = \frac{1}{\mu}\left[1 + \frac{\rho}{2(1-\rho)}\right], \quad \rho = 2\lambda/\mu < 1\,.$$

Note that $\mathbb{E}[R]$ is the forward transit time and hence, is constant, *i.e.*, $\mathbb{E}[R] = R$. Therefore, in order to minimize $\mathbb{E}[T]$ we have to minimize $\mathbb{E}[W]$. This is exactly the aim of the bus-schedule-aware policy.

The key to the superiority of the bus-schedule-aware scheduling policy is that it takes into account the *path quality* when making decisions. The path quality in the above example is the time to next bus departure. The bus-schedule-aware policy belongs to the family of scheduling policies that are commonly referred to as *opportunistic scheduling*. Opportunistic scheduling is well studied in the context of wireless networks, where the link qualities vary across users and across time, in order to maximize system throughput under

[1]We define 'end-to-end delay' as the time from the arrival of a bundle at the proxy until the bundle is received at its destination kiosk.

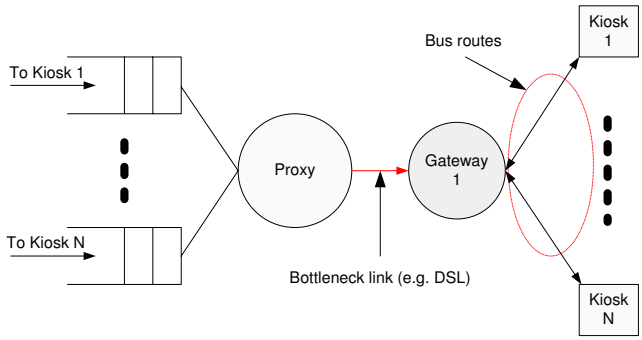

Figure 4: Each kiosk can be reached only through a single gateway.

some fairness constraints [7]. In particular, our problem is similar to opportunistic scheduling over multiple interfaces in the context of wireless networks [6, 8].

3. OPPORTUNISTIC SCHEDULING

The goal of the scheduling algorithm at the proxy is to minimize end-to-end delay while simultaneously achieving some level of fairness among different kiosks. In the following two subsections, we address the opportunistic scheduling problem for two cases. In the first case, the gateway placement is such that each kiosk can be reached from the proxy only through a single gateway. In the second case, each kiosk can be reached through multiple gateways. We refer to these two cases as *single gateway* and *multiple gateways*, respectively.

3.1 Opportunistic Scheduling: Single Gateway

Consider the network depicted in Figure 4. At each scheduling point at time t, the scheduler chooses one of the queues and schedules the head-of-line (HOL) bundle for transmission to the gateway. A scheduling point occurs whenever the gateway becomes available[2].

Consider the head-of-line bundle of queue i at scheduling point t. Let $D_i(t)$ denote the queueing delay of the HOL bundle of queue i at the proxy. Also, let $W_i(t)$ denote the time it takes the HOL bundle to reach kiosk i from the gateway at time t. Note that $W_i(t)$ consists of the time it takes for the appropriate bus to arrive at the gateway and the transit time to the kiosk. To compute $W_i(t)$, we use a modified version of Dijkstra's shortest path algorithm [3]. Therefore, if there are multiple paths to the kiosk then $W_i(t)$ is computed over the best path.

Let $Q(t) = [Q_i(t)]$ denote the scheduling policy, *i.e.*, $Q_i(t) = 1$ indicates that queue i is scheduled for transmission over the gateway. The *Single Gateway Scheduling* can be described as the following optimization problem:

$$\min_{Q(t)} \sum_{i \in N} \mathbb{E}\left[\left\{D_i(t) + W_i(t)\right\}\mathbf{I}_{Q_i(t)=1}\right], \quad (1)$$

[2]That is, the link from the proxy to the gateway becomes available.

where $\mathbf{I}$ is the indicator function define as follows,

$$\mathbf{I}_\xi = \begin{cases} 1, & \text{if } \xi = \text{true} \\ 0, & \text{otherwise} \end{cases}$$

In analogy to channel quality in wireless opportunistic scheduling algorithms, define the *path quality* $\alpha_i(t)$, $(0 \le \alpha_i(t) \le 1)$, as follows

$$\alpha_i(t) = \frac{R_i}{W_i(t)}, \quad (2)$$

where R_i is the shortest transit time from the gateway to kiosk i, *i.e.*,

$$R_i = \min_{\text{all paths at all times}} \left\{W_i(t)\right\}. \quad (3)$$

$\alpha_i(t)$ is 1 if, at this time, the ith kiosk achieves the delay minimum, and is therefore the "best" kiosk.

The proposed opportunistic scheduling policy $Q^*(t)$ is described as follows:

$$Q^*(t) = \arg\max_i \alpha_i(t). \quad (4)$$

This policy chooses the kiosk that has the best path quality at time t among all the kiosks. This ensures that a kiosk whose bus is the closest to the gateway, relative to its transit time, gets higher priority. Since the proxy system is stable, all queues will be finite and hence, fairness is achieved.

3.2 Opportunistic Scheduling: Multiple Gateways

Consider the network depicted in Figure 2. Assume that all links from the proxy to gateways have the same capacity, *i.e.*, $\mu_j = \mu, j = 1, \ldots, M$ and that at each scheduling point, all M gateways are scheduled regardless of their availability. This is equivalent to assuming that all gateways become available within a short period equal to the transmission time of a bundle. We refer to each period as a *scheduling round*. A new scheduling round begins as soon as gateways become available again.

Let $q_i(t)$ denote the number of bundles in queue i. Define $b_i(t)$ as

$$b_i(t) = \min\{q_i(t), M\}. \quad (5)$$

A little thought shows that it is possible to avoid head-of-line blocking, if the scheduling algorithm considers first $b_i(t)$ bundles at the head of queue $i = 1, \ldots, N$. We call these bundles *ready bundles*. Let $\mathcal{B}(t)$ denote the set of all ready bundles. The scheduler at time t computes an assignment from $b(t) = \sum_{i=1}^{N} b_i(t)$ ready bundles to M gateways in order to maximize the sum of scheduling gains.

Let $Q(t) = [Q_{rj}(t)]$ denote the scheduling policy, *i.e.*, $Q_{rj}(t) = 1$ indicates that bundle $r \in \mathcal{B}(t)$ is scheduled for transmission over gateway j. The *Multiple Gateways Scheduling* can be described as the following optimization problem:

$$\min_{Q(t)} \sum_{r \in \mathcal{M}(t)} \sum_{1 \le j \le M} \mathbb{E}\left[\left\{D_{rj}(t) + W_{rj}(t)\right\}\mathbf{I}_{Q_{rj}(t)=1}\right]. \quad (6)$$

Let $\alpha_{ij}(t)$ denote the path quality of kiosk i through gateway j at time t. Then

$$\alpha_{ij}(t) = \frac{R_i}{W_{ij}(t)}, \tag{7}$$

where R_i is the shortest possible delay expresses as

$$R_i = \min_{\text{all paths, gateways, time}} \Big\{ W_{ij}(t) \Big\}. \tag{8}$$

What is different from the single-gateway case is that now we need to decide for each kiosk queue which gateways we would like to use to drain that queue. After that, if more than one kiosk tries to use the same gateway, we need to resolve the conflict. We call the first step *gateway selection*, and the second *conflict resolution*.

Gateway selection is a tradeoff between "select the best" and "select all that is out there". In particular,

1. It may not be desirable to use all possible gateways to drain a kiosk queue, although that would provide the maximum instantaneous throughput. If a gateway is too far away from the destination kiosk, it may be better off not to use this gateway at all even if that means the kiosk cannot send any bundle in the current round, because later a better gateway may be scheduled to this kiosk. The greater reduction in the transit time from the gateway to the kiosk warrants a small extra queueing delay at the proxy.
2. Only utilizing the best gateway for each kiosk queue is also not desirable. The system may become unstable if the aggregate traffic of some kiosks whose choice of gateways frequently coincide exceeds the capacity of the chosen gateway. It may also occur that the traffic of a particular kiosk exceeds the capacity of any gateway. In that case, obviously using only one gateway is not sufficient to keep the system stable.

Therefore, the gateway selection strategy must be adaptive to the current load condition. For each kiosk i, we propose to use a binary function $\phi\left(\alpha_{ij}, q_i\right)$ that takes the path quality of a gateway j and the current queue length q_i as input and outputs 0 or 1 to indicate whether or not it accepts gateway j. This function is of the following general form:

$$\phi\big(\alpha_{ij}(t), q_i(t)\big) = \begin{cases} 1, & \text{if } \psi\big(\alpha_{ij}(t), q_i(t)\big) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $\psi\left(\alpha_{ij}, q_i\right)$, like $\alpha_{ij}(t)$, is a function whose range is between 0 and 1, indicating the relative "goodness" of gateway j to kiosk i, and θ is a threshold.

Intuitively, θ is the lowest path quality acceptable from kiosks perspective. That is, θ is a cut-off threshold to choose good paths. If θ is close to 0 then the scheduling algorithm is to more tolerant about available paths and hence, may choose to send bundles over paths that are not good. On the other hand, if θ is close to 1 then the scheduling algorithm is more selective and hence, may ignore some good paths. Both overly-tolerant and overly-selective algorithms will suffer from high end-to-end delays. Clearly, there is an optimal value of θ that achieves minimum delay. We will study the impact of threshold θ on end-to-end delay in Section 4.

For the function $\psi(\alpha, q)$, we believe that the exact form is not important as long as it satisfies the following properties:

$$0 \leq \psi(\alpha, q) \leq 1, \tag{i}$$
$$\psi(\alpha > 0, \infty) = 1, \tag{ii}$$
$$\psi(1, q) = 1, \tag{iii}$$
$$\psi(\alpha, q_1) \leq \psi(\alpha, q_2), \quad \text{if } q_1 \leq q_2 \tag{iv}$$
$$\psi(\alpha_1, q) \leq \psi(\alpha_2, q), \quad \text{if } \alpha_1 \leq \alpha_2 \tag{v}$$

Here is a description of what each property means:

(i) The value returned by ψ is a normalized quantity between 0 and 1, inclusive.

(ii) If the queue length is infinite then any possible path is guaranteed to be selected. Having $\alpha > 0$ indicates that the gateway can be used to reach the kiosk. This property ensures that all queues remain stable.

(iii) The best path is always selected, no matter what the queue length is.

(iv) Function ϕ is monotonically increasing with respect to the queue length q.

(v) Function ϕ is monotonically increasing with respect to the path quality α.

For our simulations, we implemented $\psi\big(\alpha_{ij}(t), q_i(t)\big)$ as follows.

1. For each pair $(\alpha_{ij}(t), q_i(t))$ compute $\omega_{ij}(t)$ as
$$\omega_{ij}(t) = 1 - e^{-\alpha_{ij}(t) q_i(t)}. \tag{11}$$
2. Compute $\psi\big(\alpha_{ij}(t), q_i(t)\big)$ as
$$\psi\big(\alpha_{ij}(t), q_i(t)\big) = \frac{\omega_{ij}(t)}{\max_{1 \leq k \leq M} \omega_{ik}(t)}. \tag{12}$$

Now, for all ready bundles $r \in \mathcal{B}(t)$ that belong to queue i, define

$$\begin{aligned} \phi_{rj}(t) &= \phi\big(\alpha_{ij}(t), q_i(t)\big), \\ \alpha_{rj}(t) &= \alpha_{ij}(t), \\ q_r(t) &= q_i(t). \end{aligned}$$

For conflict resolution, we propose the following opportunistic scheduling policy:

$$Q^*(t) = \mathcal{M}_{\arg\max_{r,j}} \Big\{ \alpha_{rj}(t) \Big\}, \tag{13}$$

where $\mathcal{M}_{\arg\max_{r,j}(.)}$ is a weighted bipartite graph matching. The nodes from one part of the graph represent ready

bundles and the nodes from another part represent the gateways. There is an edge between bundle r and gateway j if $\phi_{rj}(t) = 1$. The weight of this edge is specified by $\alpha_{rj}(t)$.

Given that buffers at the proxy are infinite and the system is stable, the long-term throughput of a kiosk will be always equal to its arrival rate. Therefore, throughput fairness is trivially achieved. We defer delay fairness to future work.

4. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our route selection and scheduling algorithm using simulation. We developed a custom simulator which proceeds in steps, where the proxy schedules bundles for transmission at the beginning of each step. The number of bundles that arrive to each kiosk is drawn from a Poisson distribution.

As a first step, we only simulate simple scenarios. That can help us further understand the problem. Performance evaluation for more complex scenarios is left as future work. Throughout this section, we will consider scenarios with two gateways and two kiosks. From each gateway, there is a separate bus to each kiosk, so there are 4 buses in total. One simulation step corresponds to approximately one minute in reality. All bundles are assumed to have fixed size. The link capacity of all proxy-to-gateway links is 1 bundle per step. So if in reality the link capacity is 100 Kbps, it implies a bundle size of approximately 750 KB. All data points are obtained by running the same simulation ten times, each time for a length of 1440 simulation steps (corresponding to 24 hours). For each point, 95% confidence intervals are shown on the plot (although most of them are too small to see). As mentioned earlier, we use (12) as our $\phi(\alpha_{ij}, q_i)$ function.

Scenario 1 (NNFF)

In this scenario, gateway g_1 is close to both kiosks and gateway g_2 is far from both kiosks. We call this scenario *Near-Near-Far-Far (NNFF)*. In particular, bus $b_{i,1}$ (that is, bus leaving from gateway g_1 serving kiosk k_i) has round trip time 60 steps, and bus $b_{i,2}$ has round trip time 120 steps. We also intentionally make bus $b_{1,j}$ and $b_{2,j}$ completely out of phase (when bus $b_{1,j}$ is at gateway g_j, bus $b_{2,j}$ is farthest away from g_j, and vice versa), to see if we can reduce overall delay by taking timing into account.

In this scenario, both kiosks should take advantage of gateway g_1 as much as possible, and use gateway g_2 only when:

1. bus $b_{i,2}$ is about to leave at gateway g_2 and the time before the next departure of bus $b_{i,1}$ from gateway g_1 is long, or,
2. when a kiosk's queue at the proxy tends to grow out of bound.

Our simulation experiments show that the proposed opportunistic scheduling policy behaves exactly in this way.

We compare our bipartite weighted matching-based algorithm (*BWM*) with a naive algorithm (*RANDOM*) that randomly assign gateways to kiosk queues. different route

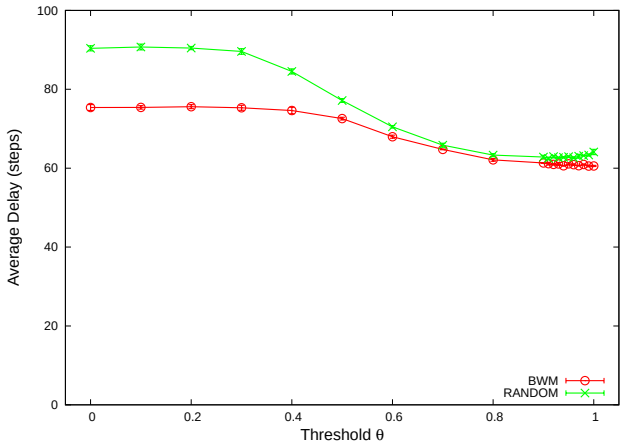

Figure 5: *NNFF* Scenario: arrival rates to both kiosks are 0.45 bundles per step.

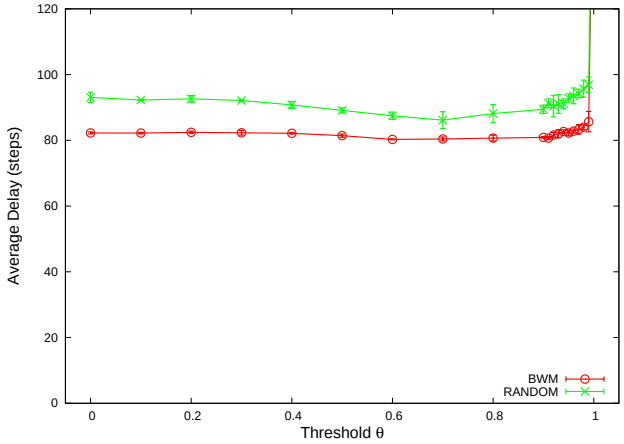

Figure 6: *NNFF* Scenario: arrival rates to both kiosks are 0.9 bundles per step.

selection and matching schemes schemes. We are also concerned with how the choice of the threshold θ in 9 will affect the performance of our algorithm. Recall that a low θ represents a gateway selection strategy that is more tolerant whereas a high θ embodies a more selective one. In particular, $\theta = 0$ corresponds to a "select-all" strategy that tries to use all gateways from which the destination kiosk is reachable, and $\theta = 1$ corresponds to a shortest-path-only strategy.

We first set the mean arrival rates to both kiosks to be 0.45 bundles per step. With these arrival rates, one single gateway is enough to serve both kiosks. Figure 5 shows the performance of *BWM* with varying threshold values and compared with *RANDOM*. We can see that *BWM* consistently performs better than *RANDOM*. For our algorithm, the best performance occurred when $\theta = 1$, indicating a shortest-path-only strategy is most desired in this setting. The average delay at this point is over one third less than achieved by *RANDOM* with *select-all*, the most naïve combination. We also notice that our algorithm has almost constant delay when θ is in the range from 0.8 to 1.

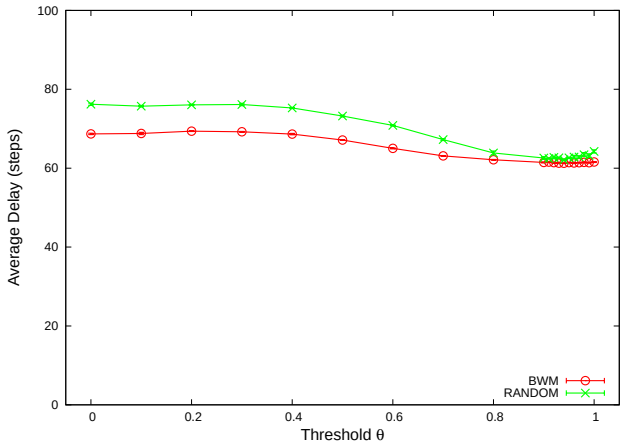

Figure 7: *NFFN* Scenario: arrival rates to both kiosks are 0.7.

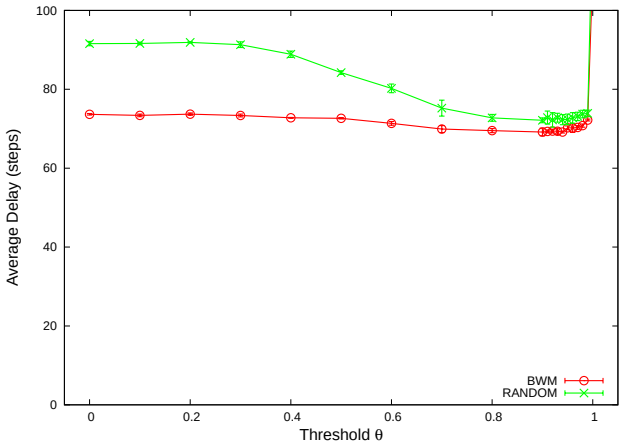

Figure 8: *NFFN* Scenario: arrival rate to kiosk k_1 is 1.2 and to kiosk k_2 is 0.5.

When compared at the same threshold value, *BWM* outperforms *RANDOM* considerably when the threshold is low. For example, when $\theta = 0$, *BWM* is about 15% better than *RANDOM*. However, the improvement becomes marginal when θ approaches 1. This is expected because when θ is small, the input bipartite graph to the conflict resolution phase is a strongly connected one, leaving more room for optimization, whereas when θ is close to 1, there are very few edges in the input bipartite graph to begin with, so any algorithm will likely perform similarly.

Although $\theta = 1$ performs the best in the previous case, it cannot adapt to changing load as it allows every kiosk to compete for only one gateway each round. To illustrate, we set the mean arrival rates for both kiosks to be 0.9 bundles per step, so that both kiosks must try to utilize gateways other than the best one to remain stable. The result is shown in Figure 6. The sudden jump at the right end of the plot clearly shows that the system becomes unstable when $\theta = 1$. When $\theta < 1$, *BWM* again consistently outperforms *RANDOM*. *BWM* has almost constant delay across a wide range of threshold values, only starting to increase when $\theta = 0.91$. This, combined with the result from Figure 5, indicates that a threshold value between 0.8 to 0.9 may be optimal for the ϕ function we use. In order to verify this conclusion, we simulate our algorithm in a different scenario.

Scenario 2 (NFFN)

Now we consider a scenario where gateway g_1 is close to kiosk k_1 but far from kiosk k_2 and gateway g_2 is close to kiosk k_2 but far from kiosk k_1. We call this scenario *Near-Far-Far-Near (NFFN)*. In particular, the round-trip time of bus $b_{1,1}$ and bus $b_{2,2}$ is 60, and the round-trip time of bus $b_{1,2}$ and bus $b_{2,1}$ is 120.

We first set the mean arrival rates to both kiosks to be 0.7 bundles per step. Note that with these arrival rates it is possible for each kiosk to use only the gateway that is close to it. Figure 7 shows the result. In Figure 8 we set the arrival rate to kiosk k_1 to be 0.5 bundles per step, and that of k_2 to be 1.2 bundles per step. This forces kiosk k_2 to use more than one gateway otherwise its queue will become unstable.

Once again, we see that *BWM* performs better than *RANDOM* in both cases. The shortest-path-only strategy performs the best in Figure 7, but is vulnerable to changing load, as illustrated by the sudden jump at the right end of Figure 8. In both cases, a threshold value between 0.8 and 0.9 allows our algorithm to achieve minimum delay.

Remarks

The simulation experiments show that our scheme with a threshold value between 0.8 and 0.9 consistently achieves the shortest delay. *Shortest-path-only* is effective in reducing delay when traffic load is light. However, it has limited stability region. *Select-all*, on the other hand, has the maximum stability region, but increases delay unnecessarily by sending bundles over longer paths. Our queue length dependent gateway selection algorithm, when supplied with an appropriate threshold (θ), has the same performance as *shortest-path-only* when the load is light, yet has a stability region as large as that of *select-all*. The bipartite weighted matching algorithm for conflict resolution consistently outperforms a naïve random matching algorithm, especially when θ is small. However, when θ is within the optimal region, the improvement is only marginal. We attribute this to the small scale of the simulated scenarios, where, when θ is large, each kiosk will compete for only 1 gateway in most cases, leaving very little room for optimization. We believe that in larger-scale scenarios where each kiosk competes for more gateways, our algorithm will exhibit more improvement over naïve algorithms.

With these encouraging initial results, we are fully aware of the simplistic nature of this simulation study. We will investigate the performance of our algorithm in more complex scenarios and under various traffic models in the future.

5. RELATED WORK

Opportunistic scheduling has been studied mostly in the context of wireless data networks. In wireless networks, the link quality fluctuates over time due to shadowing, fading, user mobility, etc. It has been shown that fluctuations in

link quality can be exploited to maximize system throughput [5]. Various fairness requirements have been studied in conjunction with opportunistic scheduling. In CDMA Ev-Do systems [1], for example, *proportional fair* scheduler [4] is applied to achieve proportional fairness among users. References [6] and [8] propose more general frameworks in which the scheduling problem is formulated as an optimization problem where the objective function is the system throughput and fairness requirements form the optimization constraints. Various fairness requirements can be expressed, including processor-sharing type of fairness.

6. CONCLUSION

In this paper, we addressed the downlink scheduling problem in ferry-based networks. We argued that by taking path qualities into account, when making scheduling decisions, the average end-to-end delay can be reduced. We then proposed opportunistic scheduling algorithms aimed at minimizing the end-to-end delay for single gateway and multiple gateway networks. Through simulations, we showed that the proposed opportunistic scheduling algorithm outperforms the non-opportunistic algorithms in reducing end-to-end delay.

We plan to tackle the following problems in future work:

- To investigate the impact of imprecise bus schedules. In practice, bus schedules can not been known precisely as assumed in this paper. Intuitively, if the degree of impreciseness is low compared to round trip times, we conjecture that the proposed heuristics should work well.
- To investigate the optimality of the proposed scheduling policies. The existing results on opportunistic scheduling policies in wireless networks [6–8] are expected to apply to our problem.
- To extend the simulation results to more realistic scenarios. We can easily simulate complicated ferry-based networks with the simulator we have developed. It will be interesting to model a real bus network in the simulator.
- To consider unreliable paths. In this case each bundle may need to be transmitted over multiple paths in order to ensure reliable delivery to its destination. It is a challenging problem to optimally choose a set of paths for each bundle to minimize delay and maximize reliability [9].
- To consider fairness when buffers at the proxy are finite. Typically, there is an admission control mechanism at the proxy that limits the number of admitted bundles. In this case, establishing fairness among multiple kiosks is a challenging problem.

7. REFERENCES

[1] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi. CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users. *IEEE Communications Magazine*, 38(7):70–77, July 2000.

[2] K. Fall. A delay tolerant networking architecture for challenged internets. In *Proc. SIGCOMM'03*, August 2003.

[3] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant networking. In *Proc. SIGCOMM'04*, August 2004.

[4] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, January 1997.

[5] R. Knopp and P. Humblet. Information capacity and power control in single cell multiuser communications. In *Proc. IEEE ICC'95*, Seattle, USA, June 1995.

[6] S. S. Kulkarni and C. Rosenberg. Opportunistic scheduling for wireless systems with multiple interfaces and multiple constraints. In *Proc. ACM Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 11–19, San Diego, USA, 2003.

[7] X. Liu, E. K. P. Chong, and N. B. Shroff. A framework for opportunistic scheduling in wireless networks. *Computer Networks*, 41(4):451–474, March 2003.

[8] Y. Liu and E. Knightly. Opportunistic fair scheduling over multiple wireless channels. In *Proc. INFOCOM'03*, San Francisco, USA, August 2003.

[9] P. Papadimitratos, Z. J. Haas, and E. G. Sirer. Path set selection in mobile ad hoc networks. In *Proc. ACM Mobihoc*, pages 1–11, Lausanne, Switzerland, 2002.

[10] K. Scott and S. Burleigh. Bundle protocol specification. draft-irtf-dtnrg-bundle-spec-04.txt, May 2006.

[11] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S.Keshav. Low-cost communication for rural Internet kiosks using mechanical backhauls. In *Proc. ACM Mobicom*, Los Angeles, USA, September 2006.