Energy-Efficient Mobile Data Transport via Online Multi-Network Packet Scheduling

Aaron Coté, Adam Meyerson, Brian Tagiku
Department of Computer Science
University of California, Los Angeles
Los Angeles, California, USA
{acote,awm,btagiku}@cs.ucla.edu

***Abstract*—We explore a novel online packet scheduling model related to energy-efficiency in mobile data transport. This model incorporates multiple networks with non-persistent connectivities where we only know which networks are available in the current timestep. When a packet arrives, it specifies a deadline and, for each network, a value it is worth if sent over that network. Our goal is to maximize the total value of packets we send by their deadlines. To encourage energy-efficiency, our model requires that packets have larger values for more energy-efficient networks. We demonstrate low-constant-competitive algorithms for this problem and several restrictions. We also provide lower bounds which closely match our competitive ratios and, under some restrictions, are tight.**

***Keywords*-packet scheduling; mobile devices; energy-efficiency; online algorithms**

I. Introduction

The introduction of the smartphone has been a boon to modern productivity and personal entertainment. They provide telephony services, internet connectivity and powerful computing all in a portable package. It is for this reason that they are quickly gaining popularity in the global mobile device market [1]. In fact, smartphones are projected to dominate the U.S. mobile phone market by 2011 [2]; it is quite clear that their universality is fast approaching.

Yet, all the new features offered by smartphones come at the cost of higher energy demands. Hence, energy-efficient methods are crucial to ensure prolonged battery life of the mobile device. It is particularly important that these energy-saving measures not affect the performance of the device. In this work, we address energy-efficiency in perhaps the most attractive feature of a smartphone – internet connectivity.

Smartphones typically have the ability to connect to the internet either using the cellular network (*e.g.* 3G or GSM) or using a WiFi connection. WiFi connections are faster and spend less energy per bit sent when the connection is active [3]. However, mobile devices spend most of their time with an idle network connection during which time cellular connections are more energy efficient. Additionally, cellular towers typically have much longer range than WiFi hotspots, offering a more persistent connection. For these reasons, most phones default to a cellular connection since they are idle for most of the time and require the user to select when they want WiFi connectivity.

Recent works such as [4], [5], [6] have proposed that mobile devices be more proactive in internet connectivity. These methods explore energy-efficient ways to connect to a WiFi connection when available. This opportunistic WiFi connectivity reaps the benefits of WiFi during data transfer and of cellular connections during idle periods. However, this alone is not enough to ensure optimal energy usage; we should actively try to schedule as many data transfers as possible into a timeframe when WiFi connectivity is available. Doing this would also have the beneficial side effect of alleviating alarmingly high demands on cellular networks [7], [8].

We address this scheduling problem at the network packet level using a novel variant of packet scheduling which we call 2-network packet scheduling. Packets have an associated value gained by sending over a cellular connection and a value gained by sending over a WiFi connection. These values give a numerical interpretation to the utility gained by successfully sending the packet per unit energy spent. Since WiFi connections are more energy efficient, this gives us motivation to try and save as many packets as possible for times when WiFi is available. However, packets also have an associated deadline by which they must be sent in order to receive any value at all. Thus, it is sometimes beneficial for us to send a packet over the cellular connection if there is no hope of sending it over WiFi.

Packet models are commonly used within the wireless network community, but the diligent reader might ponder the viability of an arbitrary-length task model. However, the packet based model offers a level of granularity a task model cannot offer. It allows for much better performance in the presence of shorter WiFi connections that could otherwise be missed in a task model. We additionally get quicker adjustment when a user initially crosses into a WiFi hotspot. Arguably, these boundary cases may compose only a small fraction of the overall energy footprint of the mobile device, but note this then implies the average connection (or disconnection) times to a WiFi hotspot is relatively long. In that case, we can expect that all packets from a divided task

are likely to be sent over the same network connection, thus closely mimicing a task model.

In this work, we provide a constant competitive algorithm for the above problem in general as well as under various restrictions. In addition, we consider a couple partially offline cases. The first assumes all packets are known up front, but may still have varying arrival times. This is a useful model when there are many recurring tasks in the system and few one-time tasks. The second assumes the availability of cellular and WiFi connections at each time step are known in advance, but packets still arrive in an online fashion. This is particularly applicable if the mobile device has a predictive system that can actively learn the user's daily routine (and thus predict when WiFi will be available). The findings in [9] strongly suggest that a very accurate predictor can in fact be built.

In all cases, we are able to provide close matching lower bounds and in some cases, tight lower bounds. Lastly, we show that the problem becomes extremely difficult when $k \geq 3$ networks are considered; no deterministic algorithm can be competitive and any randomized algorithm must have competitive ratio $\Omega(k)$. We are able to provide a matching $O(k)$-competitive randomized algorithm for this case.

II. Preliminaries

We start by reviewing the framework of online algorithms and competitive analysis. This will allow us to fully define our packet scheduling model and discuss in detail our main results.

A. Online Problems and Competitive Analysis

We say that a problem is *online* if the input is revealed piece-by-piece over a length of time. In contrast, an offline problem specifies the entire instance from the beginning. An online problem is usually accompanied by a restriction that at each timestep, some portion of the solution must be locked into place before the instance continues. This creates the difficulty of having to make decisions about the solution without knowing what will happen at future time steps.

Competitive analysis is typically the preferred method of online algorithm analysis. This form of analysis quantifies the algorithm's performance by examining the *competitive ratio* – the ratio of the algorithm's solution cost ALG to the optimal offline solution cost OPT. Typically, the competitive ratio is always expressed as a value of at least 1. Thus, if an algorithm has competitive ratio c in a particular problem instance, then this guarantees:

$$\frac{1}{c} ALG \leq OPT \leq c \cdot ALG.$$

Note that this holds regardless of whether the problem is a minimization problem or a maximization problem. We say that an online algorithm is *ρ-competitive* if its competitive ratio is bounded by ρ for any problem instance.

III. Related Work

In standard packet scheduling, packets arrive in an online fashion. Each packet p is specified by its arrival time a_p, deadline d_p and value v_p. At each time step t, we are to choose at most one packet p to send such that $a_p \leq t \leq d_p$. Our goal is to maximize the total value of packets sent. The online packet scheduling problem first appeared in 2001 in [10] and a deterministic lower bound of the golden ratio $\phi = \frac{1+\sqrt{5}}{2}$ was shown in [11] shortly after. The simple greedy algorithm (sending highest value packet first) is known to be 2-competitive [10]. Since then, a number of slight improvements have been made in [12], [13], [14] with the current best being a $2\sqrt{2} - 1 \approx 1.828$-competitive algorithm. Interestingly, a ϕ-competitive algorithm is known when packet deadlines are agreeable ($a_p < a_q$ implies $d_p \leq d_q$) [15]. For randomized algorithms, a lower bound of 1.25 and an algorithm achieving competitive ratio $\frac{e}{e-1}$, where e is the base of the natural log, are given in [16].

IV. Problem Definition and Results

In this work, we extend the packet scheduling model to accommodate multiple networks. In particular, we assume that there is persistent cellular network connection and an unpredictable WiFi network connection. Since the WiFi connectivity is unpredictable, we only know at the current time step whether WiFi is available. Each timestep is classified as either wireless or cellular. Packets possess two values, a cellular value c_p and a wireless value w_p which correspond to the amounts we receive when sent on the respective networks. We can send at most one packet per timestep, and our goal is still to maximize the total value we receive over the instance.

Definition 1 (2-Network Packet Scheduling)**.** *A set of packets $p = (a_p, d_p, c_p, w_p)$ arrive in an online fashion. At each time step t, we are told whether or not a WiFi connection is available and we must select at most one packet p with $a_p \leq t \leq d_p$ to send. If WiFi was available, then we receive w_p for sending this packet. Otherwise, we receive c_p. Our goal is to maximize the total value we receive throughout the instance.*

As evident by our definition, we assume that either a cellular or wireless connection is available at each timestep. Thus, we can classify each timestep as a cellular timestep or a WiFi timestep. We also assume that $c_p \leq w_p$ for all p to reflect the energy-savings associated with WiFi data transport.

In addition to the general problem above, we also consider the problem under each of the following restrictions:

- when $w_p \geq \alpha c_p$ for some fixed $\alpha \geq 1$,
- when all cellular values are equal,
- when all WiFi values are equal, and
- when both cellular values are equal and WiFi values are equal.

We also consider the problem when the entire set of packets is known in advance, which we refer to as offline packets. Note that this modification makes the problem strictly easier since we can no longer be "disrupted" by unexpected packet arrivals. However, the problem is still not trivial since we do not know exactly when we'll have connections to WiFi.

Complementary to the offline packets version is an offline connectivity variant. Here, we know exactly when we will have connections to WiFi, but packets arrive online. Again, this assumption makes the problem strictly easier than the general version of the problem since we know exactly how much a packet will be worth at each time step. Yet a degree of difficulty still exists due to packets arriving online.

We also generalize the above problem to accommodate $k \geq 3$ networks. Here, networks have a total ordering in terms of energy-efficiency. Thus at any given time we need only consider sending over the most energy-efficient network to which we have a connection.

We summarize our results for each of these problem variants in Table I. For all variants (except the generalized k-network version) we are able to give low-constant competitive algorithms and closely matching lower bounds. We are able to relate the offline connectivity problem to standard packet scheduling, which indicates some notion of difficulty in closing the gap in our results. For the k-network version, we prove that no deterministic algorithm can be competitive and that no randomized algorithm can offer a competitive ratio better than $\Omega(k)$. Appropriately, we have an $O(k)$-competitive randomized algorithm for this variant.

V. Notation

Throughout this work, we will use ALG to denote the schedule of an algorithm and OPT to denote the schedule of the optimum offline algorithm. We also abuse notation and use these to denote the actual sets of packets sent and also the total value obtained. It will be useful to distinguish cellularly-sent packets from wirelessly-sent packets. Thus, we use ALG_c and ALG_w to denote the set (and total value) of packets an algorithm sends over the cellular and wireless networks, respectively. We will similarly use OPT_c and OPT_w for the optimum offline algorithm. Lastly, we use ALG_{wc} and OPT_{wc} to denote the sum of the cellular values of packets in ALG_w and OPT_w, respectively. We summarize this notation in Table II.

VI. Using Feasible Schedules

A typical framework that will be used in this work will be to maintain *feasible schedules* throughout the execution of the algorithm. A feasible schedule is simply a scheduling of announced (but not necessarily arrived, as in the offline packets case) packets to distinct timeslots so that all scheduled packets could be sent, assuming persistent connectivity and ignoring any further packet announcements. For example, we can compute a feasible schedule by finding a maximum-weight matching of packets to time slots or by following a highest-value-first scheduling policy. These feasible schedules give a possible course of action and as such are useful in guaranteeing that we get a certain amount of profit from scheduled packets. In this section, we explore different types of feasible schedules and the implications of using them.

A. Feasible schedules used in this work

The first type of feasible schedule we consider is one that follows a most-valuable-packet-first scheduling policy. This type of feasible schedule allows a 2-competitive result for standard packet scheduling as shown in [10]. In fact, we will demonstrate multiple times throughout the paper that even when network connectivity is not persistent, we can remain 2-competitive.

Fact 1. *For standard packet scheduling, following a most-valuable-packet-first feasible schedule at each timestep gives a competitive ratio of 2.*

The second type of feasible schedule we will be using is a maximum-value feasible schedule which is constructed by computing a maximum-weight matching between packets and future timesteps. This type of feasible schedule allows for an interesting 2-competitive algorithm for standard packet scheduling: At each timestep t, compute a maximum-value feasible schedule and send the packet scheduled at the current timestep. In fact, this is a translation of the 2-competitive algorithm of [17] for the following problem:

Definition 2 (Online Max-Weight Matching with Vertex Locking)**.** *A set B of nodes is given and a set A of nodes arrives online, each node $a \in A$ arriving along with its weighted incident edges to B. We must construct a matching of A to B of (approximately) maximum weight. However, when we decide to match $a \in A$ to $b \in B$, this decision is not necessarily fixed for all time. Instead, the nodes of B lock one by one in an online manner. When a node $b \in B$ locks, it must keep its current match from A forever. If $b \in B$ is not matched when it locks, then it must remain unmatched forever. Note that we are not required to match all nodes of A or B, although unmatched nodes contribute zero to the objective.*

For the standard packet scheduling problem, B corresponds to the timeslots and A to the packets. A particular slot t locks when time t passes and we actually transmit packets. Thus, any competitive algorithm for this matching problem will give the same competitive ratio for packet scheduling. For completeness, we have included details of this result of [17] in the Appendix.

Fact 2. *For standard packet scheduling, following a maximum-value feasible schedule at each time step gives a competitive ratio of 2.*

	General	$w_p \geq \alpha c_p$	Equal c_p	Equal w_p	Equal w_p Equal c_p
Fully Online	CR: 3 LB: 2	CR: $\frac{3\alpha+1+\sqrt{\alpha^2+6\alpha+1}}{2\alpha}$ LB: 2	CR: $1+\phi$ LB: 2	CR: 2 LB: 2	CR: 2 LB: 2
Offline Packets	CR: 3 LB: 2	CR: $\frac{3\alpha+1+\sqrt{\alpha^2+6\alpha+1}}{2\alpha}$ LB: 2	CR: 2 LB: 2	CR: 2 LB: $\sqrt{2}$	CR: 1 LB: 1
Offline Connectivity	CR: 2 LB: ϕ	CR: 2 LB: ϕ	CR: 2 LB: ϕ	CR: 2 LB: ϕ	CR: ϕ LB: ϕ
Multiple Networks	LB: ∞	LB: ∞	LB: ∞	LB: ∞	LB: ∞

Table I
SUMMARY OF OUR DETERMINISTIC COMPETITIVE RATIOS (CR) AND LOWER BOUNDS (LB) FOR EACH PROBLEM CONSIDERED.

Symbol	Description
ALG	set or total value of packets sent by the algorithm
ALG_c	set or total value of packets sent over the cellular network in ALG
ALG_w	set or total value of packets sent over the WiFi network in ALG
ALG_{wc}	total cellular value of packets in ALG_w
OPT	set or total value of packets sent by the optimum offline algorithm
OPT_c	set or total value of packets sent over the cellular network in OPT
OPT_w	set or total value of packets sent over the WiFi network in OPT
OPT_{wc}	total cellular value of packets in OPT_w

Table II
SUMMARY OF NOTATION.

B. Effect feasible schedules have on other networks

Presumably, any algorithm that maintains a feasible schedule for a particular network connection is saving those packets in hopes that later times will have access to this network. Yet, saving these packets prevents them from being sent over any other network which can be potentially harmful to our performance. In this section, we show that this will not degrade our performance by much.

Consider an instance where all timesteps are cellular. Not knowing this, an algorithm maintains at each time t a feasible schedule S_t for timesteps $t+1$ onward for the WiFi network. Suppose once this algorithm kicks a packet out of the feasible schedule, it is never placed back into the feasible schedule. Surprisingly, if the algorithm sends the highest value packet not in S_t, it turns out it will still be 3-competitive on this instance. More generally:

Theorem 1. *Consider an instance where all timesteps are network B timesteps. If algorithm ALG maintains a feasible schedule S_t (for timesteps $t+1$ onward) for network A, never readmits packets to the feasible schedule once they are removed, and sends the highest valued packet not in S_t over network B on each timestep, then it is 3-competitive on this instance.*

Proof: Let X be the set of packets OPT sends but we do not over network B. For each packet $p \in X$, let x_p denote the first time at which it is not in the feasible schedule S_t and let a_p and d_p be its arrival and deadline, respectively. Note that since p was available to send throughout $[x_p, d_p]$ it must be that each packet we send during these times are at least as valuable.

We will show that for all times $t_1 \leq t_2$, the number of packets in X such that $[x_p, d_p] \subseteq [t_1, t_2]$ is at most $2(t_2 - t_1 + 1)$. By Hall's theorem, this will imply a matching in which each of our packets are matched to at most two packets in X (of equal or lesser value) and all packets in X are matched.

Fix t_1, t_2 and let $X(t_1, t_2)$ denote the set of packets described above. If p is a packet with $a_p \geq t_1$ then $[a_p, d_p] \subseteq [t_1, t_2]$. Thus, OPT must have sent p at some time within $[t_1, t_2]$. It follows that there can be no more than $(t_2 - t_1 + 1)$ of these packets. The remaining packets in $X(t_1, t_2)$ all have $a_p < t_1$. But since $x_p \geq t_1$ for all these packets, at time $t_1 - 1$ we were able to fit all these packets in our feasible schedule. Since all these packets have deadline no later than t_2, it follows that there are no more than $(t_2 - t_1 + 1)$ of these packets. Thus, in total $|X(t_1, t_2)| \leq 2(t_2 - t_1 + 1)$.

So the total value OPT gets from packets in X is no more than twice our total profit. Moreover, the total value OPT gets from packets we also send is at most our total profit. It follows that we must be 3-competitive. ■

VII. ONLINE PACKETS

We start by examining the problem in the fully-online situation defined above. We first start with the most general case, then move on towards equal cellular values, equal wireless values and equivalently valued packets.

```
for all timesteps t do
    Compute most-valuable-packet-first feasible schedule
    S_t for times t+1 onward using WiFi values
    if t is cellular then
        send highest cellular value packet not in S_t
    else
        send highest WiFi value packet
    end if
end for
```

Figure 1. Algorithm for fully online, general 2-network packet scheduling.

A. Arbitrary wireless and cellular values

We now show that in the general online problem, it is possible to be 3-competitive. Our algorithm is as shown in Figure 1. We prove that this algorithm is 3-competitive via the following lemmas:

Lemma 1. $2ALG_w \geq OPT_w$.

Proof: We define mapping $\pi : OPT_w \rightarrow ALG_w$ such that $w_p \leq w_{\pi(p)}$ for every $p \in OPT_w$ and such that for every $q \in ALG_w$, we have $|\pi^{-1}(q)| \leq 2$. The existence of such a mapping proves the claim. Consider a packet p that OPT sends over WiFi at time t_p^*. If $p \in ALG_w$ then set $\pi(p) = p$. If at time t_p^* we did not yet send p then p was a candidate packet during t_p^*. This means we must send a packet q of wireless value at least w_p at time t_p^*, so set $\pi(p) = q$. Otherwise we send p over cellular at some time $t_p < t_p^*$. But at time t_p it must be that there were packets scheduled by S_{t_p} from times $t_p + 1$ up to d_p with wireless value at least t_p. One of these packets must be available to be sent at time t_p^*. Thus, we know that the packet q we send at time t_p^* has wireless value at least w_p and we can set $\pi(p) = q$. Notice that each of our wireless packets is charged at most twice: once by itself and once by the packet OPT sends at the same time. Thus our wireless value is at most half the optimum wireless value. ∎

Lemma 2. $ALG_w + 3ALG_c \geq OPT_c$.

Proof: If at every time, we send the highest cellular value packet not in S_t then, by Theorem 1, we would have $3ALG_c \geq OPT_c$. However, at WiFi times, we may not be able to send these cellular packets. A close examination of the proof of Theorem 1 will show that these WiFi timesteps become matched with at most one packet that OPT sends cellularly but we do not. This follows simply because OPT cannot send a cellular packet during these times. Additionally, the wireless value we get is greater than the cellular value we would have received. Thus, we recover our losses if we add the value we get from all our wireless packets, finishing the proof. ∎

We can now add the inequalities of Lemmas 1 and 2 to show that our algorithm is 3-competitive:

```
for all timesteps t do
    Compute maximum-value feasible schedule S_t for
    times t onward using cellular values, breaking ties
    using WiFi values. Ensure that for each packet p
    scheduled at time t_p all packets q scheduled at times
    in [t_p, d_p] satisfy w_q − c_q ≥ w_p − c_p.
    if t is cellular then
        send the packet scheduled by S_t
    else
        p ← scheduled packet in S_t
        q ← packet with max w_q − c_q of those available
        if w_q − c_q > βw_p then
            send q
        else
            send p
        end if
    end if
end for
```

Figure 2. Algorithm for fully online, 2-network packet scheduling where wireless values are α times greater than cellular values.

Theorem 2. *The algorithm in Figure 1 is 3-competitive.*

B. Wireless values α times greater than cellular values

If we are guaranteed that for each packet p we have $w_p \geq \alpha c_p$ for some large α, then we can actually improve our competitive ratio using the algorithm outlined in Figure 2. To analyze this algorithm, we define the following additional notation: We let ALG_f denote the value ALG gets from all packets it sent wirelessly that were first in the feasible schedule at that time. On the other hand, ALG_d will denote the value ALG gets from all packets it sent wirelessly that had the highest difference at that time. Note that every wireless packet falls under exactly one of ALG_f or ALG_d. We also use ALG_{fc} and ALG_{dc} to denote the respective total cellular value of packets in ALG_f and ALG_d. Finally, we use $KICK$ to denote the total cellular value of packets that were displaced from the feasible schedule due to WiFi times.

Lemma 3. $2(ALG_d - ALG_{dc}) + (1+\beta)ALG_f - ALG_{fc} \geq OPT_w - OPT_{wc}$.

Proof: Let $D(t)$ to be the value of $2(ALG_d - ALG_{dc})$ and $F(t)$ be the value of $(1+\beta)ALG_f - ALG_{fc}$ at time t. Assuming that the packets ALG has sent up to time t are no longer available, let $Z(t)$ be the maximum sum of differences (*e.g.* $w_p - c_p$) that can be obtained at WiFi times from t onward. Note that $D(0) = F(0) = 0$ and $Z(0) \geq OPT_w - OPT_{wc}$ giving $D(0) + F(0) + Z(0) \geq OPT_w - OPT_{wc}$. Thus, it is sufficient to show that the following holds for all times t:

$$D(t+1) + F(t+1) + Z(t+1) \geq D(t) + F(t) + Z(t). \quad (1)$$

Suppose $t+1$ is a WiFi time. Then our algorithm schedules some packet p and $Z(t)$ schedules some packet q. If $p \in ALG_f$ then

$$D(t+1) = D(t)$$
$$F(t+1) = F(t) + (1+\beta)w_p - c_p$$
$$Z(t+1) \geq Z(t) - (w_q - c_q) - (w_p - c_p).$$

Since we know $\beta w_p \geq w_q - c_q$ we can combine inequalities to get Equation 1. If $p \in ALG_d$ then

$$D(t+1) = D(t) + 2(w_p - c_p)$$
$$F(t+1) = F(t)$$
$$Z(t+1) \geq Z(t) - (w_q - c_q) - (w_p - c_p)$$

Since we know that $w_p - c_p \geq w_q - c_q$ we again get Equation 1. Thus, Equation 1 holds on every wireless timestep t.

Now suppose that $t+1$ is a cellular time. In this case, our algorithm sends precisely the packet p that was scheduled in the feasible schedule, but $Z(t)$ sends nothing since it only considers WiFi timeslots. Clearly $D(t+1) = D(t)$ and $F(t+1) = F(t)$. We will show that there exists some $Z(t)$ solution that does not schedule p. This implies $Z(t+1) = Z(t)$ and that Equation 1 holds on every cellular timestep, thus completing the proof.

Suppose, by way of contradiction, that all $Z(t)$ solutions schedule p and consider the one that schedules p earliest. Note since $t+1$ is cellular and $Z(t)$ only considers WiFi timeslots, we know the time t_p at which $Z(t)$ sends p satisfies $t_p > t+1$. Thus, we have that $d_p > t+1$.

Since the algorithm sent p, it is currently the first packet in the feasible schedule, and any other packet q in the feasible schedule up to d_p must satisfy $w_q - c_q \geq w_p - c_p$. If one of these packets has deadline $d_q > d_p$, then we can make the same claim for all packets up to d_q. Ultimately, we can find some time $T \geq d_p$ such that packets are scheduled at every timestep between $t+1$ and T inclusive, all of these packets have deadline T or earlier, and each of these packets q satisfy $(w_q - c_q) \geq (w_p - c_p)$. Let S denote the set of packets the feasible schedule sends between t_p and T inclusive and note that $p \notin S$. Thus, $Z(t)$ must send some packet q in S prior to t_p. However, we can then swap p and q and obtain a $Z(t)$ that schedules p earlier, and still obey deadlines. This contradicts our assumption that $Z(t)$ sends p as early as possible. It follows that some $Z(t)$ solution must not send p. ∎

Lemma 4. $2(ALG_c + ALG_{dc} + ALG_{fc} + KICK) \geq OPT_c + OPT_{wc}$.

Proof: Let OPT' be the optimum offline solution if all time steps were cellular. Clearly $OPT' \geq OPT_c + OPT_{wc}$. By Fact 2, we know that always following our feasible schedule will keep us 2-competitive. However, certain packets may be prematurely sent due to a WiFi time slot, but these are precisely the packets in ALG_{dc}. These packets may kick out the scheduled packets at those times which comprise $KICK$. If we account for all these packets, we get our result. ∎

```
1: for all timesteps t do
2:    Compute most-valuable-packet-first feasible schedule
      S_t consisting of only those packets p with w_p ≥ φ
      for times t+1 onward using WiFi values
3:    if t is cellular then
4:       send the earliest deadline packet not in S_t
5:    else
6:       if S_t ≠ ∅ then
7:          send the highest WiFi value packet
8:       else
9:          send the earliest deadline packet
10:      end if
11:   end if
12: end for
```

Figure 3. Algorithm for fully online, 2-network packet scheduling where all cellular values are equal.

Combining Lemmas 3 and 4 give us the following theorem:

Theorem 3. *The algorithm in Figure 2 is* $\left(\frac{3\alpha+1+\sqrt{\alpha^2+6\alpha+1}}{2\alpha}\right)$*-competitive.*

Proof: The inequalities of Lemmas 3 and 4 give:

$$2ALG_c + 2ALG_d + (1+\beta)ALG_f + ALG_{fc} + 2KICK \geq OPT_c + OPT_w = OPT.$$

However, notice that for each packet in $KICK$, we sent a packet from ALG_d that was at least $\alpha\beta$ times more valuable. Moreover, $ALG_{fc} \leq ALG_f/\alpha$. This gives

$$\max\left\{1+\beta+\frac{1}{\alpha}, 2+\frac{2}{\alpha\beta}\right\} ALG \geq 2ALG_c + \left(2+\frac{2}{\alpha\beta}\right) ALG_d + \left(1+\beta+\frac{1}{\alpha}\right) ALG_f \geq OPT.$$

Optimizing β gives

$$\beta = \frac{(\alpha-1)+\sqrt{\alpha^2+6\alpha+1}}{2\alpha}$$

Thus, our competitive ratio is as claimed. ∎

We note that plugging in $\alpha = 1$ gives a competitive ratio of $(2+\sqrt{2}) \approx 3.4142$ which is slightly worse than our general algorithm. However, if $\alpha \geq \frac{3}{2}$ then we are better than 3-competitive. Moreover, as α approaches infinity, we approach competitive ratio 2.

C. Equal cellular values

If all packets have identical cellular value $c_p = 1$ then we can actually be $(1+\phi)$-competitive, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Our algorithm is given in Figure 3. To show this algorithm is $(1+\phi)$-competitive, we slightly modify our definition of ALG_w to be those packets we send over WiFi which have wireless values at least ϕ. We then define ALG_x to be those packets we send over WiFi with wireless value smaller than ϕ. We similarly define OPT_w and OPT_x.

Lemma 5. $2|ALG_c| + 2|ALG_x| + 3|ALG_w| \geq |OPT_c| + |OPT_w| + |OPT_x|$.

Proof: Consider a packet $p \in OPT - ALG_w$. Since $p \notin ALG_w$, it was removed from S_t at some time x_p which we will call p's *virtual arrival*. Using Hall's Theorem in a manner similar to the proof of Theorem 1, we can show that it is possible to schedule at least half the packets in $OPT - ALG_w$ within their respective virtual arrival times and deadlines. Since we send all these packets earliest deadline first, we will send at least $\frac{1}{2}|OPT - ALG_w|$ packets. Thus, $|ALG| \geq \frac{1}{2}|OPT - ALG_w|$ which can be rearranged to give the claim. ■

Further partition OPT_x into OPT'_x and OPT''_x where OPT'_x will be the packets OPT sends at a time when we send a packet in ALG_w and $OPT''_x = OPT_x - OPT'_x$.

Lemma 6. $2ALG_w - 2|ALG_w| \geq OPT_w - |OPT_w| + OPT'_x - |OPT'_x|$.

Proof: In a manner similar to the proof of Lemma 1, we can construct a mapping $\pi : OPT_w \cup OPT'_x \rightarrow ALG_w$ such that for each $p \in OPT_w \cup OPT'_x$ we have $w_{\pi(p)} \geq w_p$ and such that for every $q \in ALG_w$ we have $|\pi^{-1}(q)| \leq 2$. Since all cellular values are equal to 1 and all wireless values are at least 1, the lemma follows. ■

Lemma 7. $(\phi - 1)|ALG_c| + (\phi - 1)|ALG_x| \geq OPT''_x - |OPT''_x|$.

Proof: Notice that all times at which OPT sends packets in OPT''_x are times during which we follow an earliest deadline first scheduling policy. Since the packets of OPT''_x are never in S_t, we must send at least $|OPT''_x|$ packets during the times we follow an earliest deadline first scheduling policy. Notice that the packets we send during these times are precisely those in ALG_c or ALG_x, so $|ALG_c| + |ALG_x| \geq |OPT''_x|$. Clearly, $\phi|OPT''_x| \geq OPT''_x$ so $(\phi - 1)|OPT''_x| \geq OPT''_x - |OPT''_x|$. Combining both inequalities proves the claim. ■

Lemma 8. *The algorithm in Figure 3 is* $(1+\phi)$*-competitive.*

Proof: Combining the right-hand sides of lemmas 5, 6 and 7 gives precisely OPT. The left-hand sides sum to

$$(1 + \phi)(|ALG_c| + |ALG_x|) + 2ALG_w + |ALG_w|.$$

1: **for all** timesteps t **do**
2: Compute maximum-value feasible schedule S_t for times t onward using cellular value. Ensure that for each packet p scheduled at time t_p, all packets q scheduled at times $[t_p, d_p]$ have deadline at least d_p.
3: **if** S_t schedules packet at time t **then**
4: send the packet scheduled by S_t
5: **else if** t is a WiFi time **then**
6: send the packet with earliest deadline
7: **end if**
8: **end for**

Figure 4. Algorithm for fully online, 2-network packet scheduling where all WiFi values are equal.

Since all cellular values are 1, $ALG_c = |ALG_c|$. We obtain value at least 1 for each ALG_x packet, so $ALG_x \geq |ALG_x|$. Finally, $(\phi - 1)ALG_w = \frac{1}{\phi}ALG_w \geq |ALG_w|$. Thus, the left-hand sides sum to no more than $(1 + \phi)ALG$. ■

D. Equal wireless values

We now consider the case where all packets have identical wireless value W, but may have varying cellular values (less than W). We can show that the algorithm in Figure 4 is in fact 2-competitive.

Lemma 9. $ALG_w \geq OPT_w$.

Proof: Let $ALG_w(t)$ denote the wireless value obtained by our algorithm up to time t and $Z(t)$ denote the optimum offline wireless value obtainable from all times after t given the decisions of our algorithm up to time t. Note that $ALG_w(0) + Z(0) = OPT_w$. We will show that $ALG_w(t) + Z(t)$ is non-decreasing, proving the claim since at the final time t_f we have $Z(t_f) = 0$.

Suppose $t+1$ is a cellular time. Then clearly $ALG_w(t+1) = ALG_w(t)$. Suppose our algorithm decides to send some packet p. Let T be the first time slot at which the current feasible schedule has no packet scheduled. Let S be the set of packets currently scheduled between time $t+1$ and T. Note that since we send p, it must be that $T \geq d_p$ and for each $q \in S$ we have $d_p \leq d_q \leq T$. Suppose, by way of contradiction, that all $Z(t)$ solutions send p and consider the one that schedules p earliest. Let t_p be the time $Z(t)$ schedules p and note $t_p > t$. No packet in S can be sent by $Z(t)$ over wireless prior to t_p otherwise we could swap with p and get an earlier time for p. However, this means some packet in S must not be sent wirelessly by $Z(t)$ and we could swap with p to get a schedule $Z(t)$ that doesn't use p. Thus $Z(t+1) = Z(t)$.

Now, suppose $t+1$ is a WiFi time. If our algorithm is unable to send a packet, then clearly $ALG_w(t+1) = ALG_w(t)$ and $Z(t+1) = Z(t)$. If our algorithm sends some packet p but $Z(t)$ has no packet scheduled, then $ALG_w(t+1) = ALG_w(t) + W$ and $Z(t+1) \geq Z(t) - W$.

If our algorithm sends p and $Z(t)$ sends q then we know $d_p \leq d_q$. If $Z(t)$ scheduled p then we could swap p and q and have $Z(t)$ send p at this time instead. Otherwise, if $Z(t)$ doesn't schedule p, we have $ALG_w(t+1) = ALG_w(t) + W$ and $Z(t+1) \geq Z(t) - W$. In all cases, we have $ALG_w(t+1) + Z(t+1) \geq ALG_w(t) + Z(t)$. ■

Lemma 10. $2ALG_c + ALG_w \geq OPT_c$.

Proof: Following the Online Maximum-Weighted Matching with Vertex Locking proof in [17], let Δ_a denote the increase in matching weight upon arrival of task a. Since we occasionally send scheduled packets over WiFi, we now get $ALG_w + ALG_c \geq \sum_a \Delta_a$. Let $\rho_t(b)$ denote the decrease in matching weight at time t if we remove time slot b as a potential match. Thus, if $\rho(b)$ represents the final $\rho_t(b)$, then we have $\sum_{b \text{ cellular}} \rho(b) = ALG_c$. Note that $\rho_t(b)$ is still non-decreasing with time for cellular time slots b. Our proof of this holds when packets arrive and when time slots lock, so we need only show $\rho_t(b)$ is non-decreasing when a time slot is declared WiFi. Note that when a time slot is declared WiFi the bipartite graph remains unchanged. Thus, any matching that minimizes b's matched value at this new time was also possible before WiFi was declared. So it follows that $\rho_t(b) \leq \rho_{t+1}(b)$ and we get equality when b is *not* the slot being declared WiFi.

Thus, for each a and cellular b, we must have $\Delta_a + \rho_t(b) \geq w_{ab}$ where w_{ab} is the weight of matching a to b. Summing over all OPT edges in the matching proves the claim. ■

We can simply combine Lemmas 9 and 10 to get the following:

Theorem 4. *The algorithm in Figure 4 is* 2*-competitive.*

VIII. Offline Packets

We now consider the variant in which all packets are known up front (but may still have varying arrival times). Under this model, we improve our competitive ratios for the online packet model when cellular values are equal. In the case where all wireless values are equal and all cellular values are equal, we show we can be optimal.

A. Equal cellular values

We will show that the algorithm in Figure 5 is 2-competitive. This gives a slight improvement over our result for the fully online version of this problem.

Lemma 11. $2ALG_w - 2ALG_{wc} \geq OPT_w - OPT_{wc}$.

Proof: Let p be a packet OPT sends over WiFi at time t_p^*. If $p \in ALG_w$ then set $\pi(p) = p$. If at time t_p^* we did not yet send p then p was a candidate packet during t_p^*. This means we must send a packet q of wireless value at least w_p at time t_p^*, so set $\pi(p) = q$. Otherwise we send p over cellular at some time $t_p < t_p^*$. But at time t_p it must be that there were packets scheduled by S_{t_p} from times $t_p + 1$ up to d_p with wireless value at least t_p. One of these packets must be available to be sent at time t_p^*. Thus, we know that the packet q we send at time t_p^* has wireless value at least w_p and we can set $\pi(p) = q$. Notice that each of our wireless packets is charged at most twice: once by itself and once by the packet OPT sends at the same time. Thus our wireless value is at most half the optimum wireless value. Moreover, since all packets have the same cellular value, the claim follows. ■

```
for all timesteps t do
    Compute most-valuable-packet-first feasible schedule
    S_t for times t+1 onward using WiFi values.
    if t is cellular then
        send the earliest deadline packet not in S_t
    else
        send the highest wireless value packet
    end if
end for
```

Figure 5. Algorithm for 2-network packet scheduling with offline packets where all cellular values are equal.

Lemma 12. $2ALG_{wc} + 2ALG_c \geq OPT_c + OPT_{wc}$.

Proof: Note that at time $t = 0$, we are guaranteed to get at least the cellular value of all the packets in S_0. Then if $OPT_c(S_0)$ denotes the total value of packets in S_0 which optimum sends at cellular and $OPT_{wc}(S_0)$ denote the total cellular value of packets in S_0 which optimum sends over WiFi, we have

$$ALG_c + ALG_{wc} \geq OPT_c(S_0) + OPT_{wc}(S_0).$$

At time $t = 0$, we are also guaranteed to get at least the optimum cellular value of packets not in S_0 since earliest deadline first is optimal in this case. Thus, we must have

$$ALG_c \geq OPT_c(\overline{S_0}).$$

Finally, consider all those packets in $\overline{S_0}$ that optimum sends over WiFi. Since these packets are not in S_0, we must have a packet scheduled at every time during their windows. Thus, whenever OPT sends one of these packets, we also send a packet. Thus, we have

$$ALG_{wc} \geq OPT_{wc}(\overline{S_0}).$$

Combining all the inequalities proves the claim. ■

Theorem 5. *The algorithm in Figure 5 is 2-competitive.*

B. Equal wireless and equal cellular values

We show that when all packets have equal wireless value and equal cellular value we can be optimal. This algorithm is as shown in Figure 6.

Theorem 6. *The algorithm in Figure 6 is optimal.*

Proof: We first show that we send a wireless packet whenever OPT does. Let $p \in OPT_w$ and let t be the time

for all timesteps t **do**
  Compute maximum-value feasible schedule S_t for times t onward using cellular values. Ensure that for each packet p scheduled at time t_p, there are packets scheduled every time between t_p and d_p with deadline no earlier than d_p.
  if S_t schedules packet at time t **then**
    send the packet scheduled by S_t
  else if t is WiFi **then**
    send the packet with earliest deadline
  end if
end for

Figure 6. Algorithm for 2-network packet scheduling with offline packets where all cellular values are equal and all WiFi values are equal.

OPT sent it. If we did not yet send p at time t, then it is clear we must send some packet at t. If we already sent p prior to t, then we know there were packets scheduled up until d_p. Thus, one of these packets must be available for us to send. It follows that

$$ALG_w - ALG_{wc} \geq OPT_w - OPT_{wc}.$$

Note that all packets scheduled at time $t = 0$ are eventually sent in some manner. Since we originally scheduled the maximum number of packets possible, it follows that

$$ALG_c + ALG_{wc} \geq OPT_c + OPT_{wc}.$$

Putting inequalities together gives $ALG \geq OPT$ and proves the claim. ■

IX. Offline Connectivity

We now consider the variant in which network connectivity is known in advance throughout the entire instance, but packets arrive in an online fashion. Again, this model is strictly easier than the fully online model and, consequently, we can improve our competitive ratios. In particular, we improve our competitive ratio for the general problem and for the case when all packets have equal cellular value and WiFi value.

A. Arbitrary wireless and cellular values

Since all connectivities are known in advance, we know precisely what each (arrived) packet's value will be at each future timestep. Thus, we can simply employ the matching algorithm of [17] to immediately get a competitive ratio of 2. The algorithm works simply by, at each timestep, computing a maximum-value matching of packets to timesteps and sending the packet matched to the current timestep.

Theorem 7. *The above algorithm is 2-competitive.*

B. Equal wireless and equal cellular values

If all packets have equal wireless value W and equal cellular value C, then we can do slightly better than above. Note that if $\frac{W}{C} < \phi$ then for every packet we send, we always obtain $\frac{1}{\phi}$ of the value OPT could receive from this packet. Thus, we need only make sure we send more packets than OPT. This can be achieved by simply using an earliest-deadline-first scheduling policy, regardless of what network connectivity we have.

On the other hand, if $\frac{W}{C} \geq \phi$ then the following policy is ϕ-competitive: On a WiFi timestep, we simply send the packet with earliest deadline. On a cellular timestep, we construct a maximum matching of packets to WiFi timesteps. We then find the packet p of earliest deadline whose removal still allows the same number of WiFi timesteps to be matched. If such a p exists we send it, otherwise we send nothing.

Our algorithm then runs the first policy if $\frac{W}{C} < \phi$ and runs the second policy if $\frac{W}{C} \geq \phi$.

Theorem 8. *The above algorithm is ϕ-competitive.*

Proof: It is clear that when $\frac{W}{C} < \phi$ the first policy achieves competitive ratio of ϕ. Thus, we need only show that the second policy is ϕ-competitive when $\frac{W}{C} \geq \phi$. Clearly, since we maximize the number of WiFi packets we can send, we have $ALG_w \geq OPT_w$.

If we had ignored WiFi timesteps entirely and followed an earliest-deadline-first policy, then we would also collect optimum cellular value. However, there may be select cellullar times when OPT sends something and we do not because we are saving it for a WiFi timestep. In this case, $\frac{1}{\phi}$ of the value collected from this WiFi timestep is enough to cover the cellular value we've lost. Thus, we have $\frac{1}{\phi} ALG_w + ALG_c \geq OPT_c$. Combining this with the above inequality proves the theorem. ■

X. Lower Bounds for 2 Networks

We now prove lower bounds on the competitive ratio of any deterministic algorithm. We start by showing that when packets and connectivity are online, no deterministic algorithm can have competitive ratio better than 2, even under the restriction that all cellular values are equal and all wireless values are equal. Thus this gives a lower bound for all versions with online packets.

Theorem 9. *When connectivity is online, packets arrive online and all packets have equal cellular value and equal wireless value, no deterministic algorithm can be c-competitive for $c < 2$.*

Proof: Let ALG be a deterministic algorithm with competitive ratio c. We build a bad instance for this algorithm as follows: Consider starting at $t = 0$ with a packet $A = (0, 1, 1, 2)$ and cellular connectivity. If the algorithm decides to send A at this time, then at time $t = 1$ we get WiFi

connectivity. If the algorithm decides not to send A, then in the next time step a packet $B = (1, 1, 1, 2)$ arrives and we have cellular connectivity. In either case, the algorithm gets total value 1 whereas the optimum offline value is 2. It follows that $c \geq 2$. ∎

In fact, a similar example shows that no randomized algorithm can be better than $\frac{3}{2}$-competitive:

Theorem 10. *When connectivity is online, packets arrive online and all packets have equal cellular value and equal wireless value, no randomized algorithm can be c-competitive for $c < \frac{3}{2}$.*

Proof: Suppose, by way of contradiction, that we have a c-competitive randomized algorithm ALG for $c < \frac{3}{2}$. Consider an instance where at time 0 packet $A = (0, 1, 1, \alpha)$ arrives and we have cellular connectivity. The algorithm decides to send this packet with some probability p. From here, the instance can run in two ways: (1) in the next timestep we get wireless connectivity, (2) we still have cellular connectivity but a packet $B = (1, 1, 1, \alpha)$. In the first case, the algorithm's competitive ratio is $\frac{\alpha}{p+(1-p)\alpha}$ and in the second case, the competitive ratio is $\frac{2}{2p+(1-p)}$. Thus, the algorithm's ratio is best when these two quantities are equal. Solving for p gives $p = \frac{\alpha}{3\alpha-2}$ in which case our compatitive ratio is $\frac{3\alpha-2}{2\alpha-1}$ which approaches $\frac{3}{2}$ as $\alpha \to \infty$. ∎

Next, we show that when connectivity is online, packets are offline and cellular values are equal, no deterministic algorithm can be better than 2-competitive:

Theorem 11. *When connectivity is online, but packets are offline and all packets have equal cellular values, no deterministic algorithm can be c-competitive for $c < 2$.*

Proof: Let ALG be a deterministic algorithm with competitive ratio c. Our bad instance for this algorithm will have two packets $A = (0, 1, 1, 4)$ and $B = (1, 1, 1, 1)$. At time $t = 0$ we will have only cellular connectivity. If ALG sends A, then at the next timestep, we get wireless and can only get total value 2 whereas the optimum value is 4. If ALG does not send A at this time, then in the next time step we have cellular connectivity again. We can only send one of A and B whereas it was possible to send both. Since ALG must follow one of these cases, it follows that it cannot guarantee less than half the optimum profit. Thus $c \geq 2$. ∎

When connectivity is online, packets are offline and WiFi values are equal, we can show a slightly weaker lower bound:

Theorem 12. *When connectivity is online, but packets are offline and all packets have equal wireless values, no deterministic algorithm can be c-competitive for $c < \sqrt{2}$.*

Proof: Consider an instance with the following packets: $A = (0, 0, \beta, 1), B = (0, 1, 1, 1)$ and $C = (0, 2, 0, 1)$. Suppose at time 0 we have only cellular connectivity. If ALG sends A, then at $t = 1$ we'll have wireless and at $t = 2$ we'll only have cellular. Among B and C, we can only hope to get additional value 1. Thus, our total value is $1 + \beta$ whereas OPT can get 2 (by sending B at time 0 and C at time 1). Notice that a similar situation arises if the algorithm sends C at time 0 or nothing at time 0. If ALG sends B at time 0, then the remaining times are all cellular. In this case, we only get total value 1 whereas OPT could get $1 + \beta$.

Thus, our competitive ratio is the smallest of $\frac{1}{1+\beta}$ and $\frac{1+\beta}{2}$. Setting both equal gives $\beta = \sqrt{2}-1$ which corresponds to competitive ratio $\sqrt{2}$. Thus, ALG can be no better than $\sqrt{2}$-competitive. ∎

When connectivities are offline, we can show a lower bound of ϕ even if all packets have equal wireless value and equal cellular value:

Theorem 13. *When connectivities are offline and all packets have equal wireless value and equal cellular value, no deterministic algorithm can be c-competitive for $c < \phi$.*

Proof: Consider a two timestep instance where at $t = 0$ we have only a cellular connection and at $t = 1$ we have a WiFi connection. At $t = 0$ packet $A = (0, 1, 1, \phi)$ arrives. If ALG sends A, then no new packets arrive and the instance ends. Thus, ALG gets total value 1 whereas OPT could get value ϕ. If ALG does not send A at $t = 0$ then at time $t = 1$ packet $B = (1, 1, 1, \phi)$ arrives. The most ALG can get is ϕ whereas OPT could have received $1 + \phi$. In either case, ALG has competitive ratio ϕ. ∎

While there is a gap in many of the cases for offline connectivities, we can easily show that a c-competitive algorithm for this problem implies a c-competitive algorithm for standard packet scheduling. This follows simply because we can treat all timesteps as being of the same network type. Thus, this is in fact true even for the version where all cellular values are equal, or the version where all WiFi values are equal. Given the difficulty in achieving a ϕ-competitive algorithm for standard packet scheduling, this implies a level of difficulty in getting a tight bound for 2-network packet scheduling with offline connectivities.

Theorem 14. *A c-competitive algorithm for 2-Network Packet Scheduling with offline connectivities even when restricted to equal cellular values, or equal WiFi values (but not both) is a c-competitive algorithm for standard packet scheduling.*

XI. More Than 2 Networks

A natural generalization of this problem is to consider a hierarchy of network connections. In general, assume we have k types of connections but are guaranteed at least the lowest connection type will be available. Each packet p is then specified by a larger tuple $(a_p, d_p, v_{p,1}, v_{p,2}, \ldots, v_{p,k})$ where $v_{p,i}$ denotes its value if sent over network i. We'll

assume the values for each packet satisfy $v_{p,i} \leq v_{p,i+1}$ for all i.

A. Lower Bounds for $k \geq 3$ Networks

Unfortunately, we can show that no deterministic algorithm can be c-competitive for any c:

Theorem 15. *No deterministic algorithm can have competitive ratio independent of values.*

Proof: We consider instances with one packet and two time steps and three network types. Thus, this also shows that there's very little we can do to obtain superconstant competitive ratios (except possibly if we have dependence on ratio of values). The example is as follows: Suppose at time 0 we have type-2 connectivity and a packet $A = (0, 1, 1, \alpha, \alpha^2)$ arrives where $\alpha > c$. For our algorithm to be c-competitive, it cannot send the packet at this time step since the next time step could have type-3 connectivity. Similarly, it cannot delay the packet at this time step since the next time step could have type-1 connectivity. However, it must do one of the two so we have a contradiction. ■

Things are not as bad for randomized algorithms, but a large lower bound can still be demonstrated:

Theorem 16. *No randomized algorithm can be better than $(k-1)$-competitive*

Proof: We define a collection of instances $\mathcal{I}_0, \ldots, \mathcal{I}_{k-2}$ each with one packet $A = (0, k-1, 1, \alpha, \ldots, \alpha^{k-1})$ and $k-1$ timesteps. Instance $\mathcal{I}_i$ has type-j connectivity in timestep $j-2$ for $2 \leq j \leq i+2$ and type-1 connectivity everywhere else.

Suppose we have an r-competitive randomized algorithm. It must send A at time 0 with probability $p \geq \frac{1}{r}$ in order to be competitive on instance $\mathcal{I}_0$. Similarly, it must send A at time 1 with probability $p \geq \frac{1}{r}$ to be competitive on instance $\mathcal{I}_1$. Thus, it must send A with probability at least $\frac{1}{r}$ on each of the $k-1$ timesteps. Since the sum of these probabilities must be at most 1, we get $r \geq k-1$. ■

B. Randomized, $\frac{ek}{e-1}$-Competitive Algorithm

While there is no hope for a deterministic algorithm, we can give a randomized $\frac{ek}{e-1}$-competitive algorithm. This algorithm works as follows: When a packet comes in, we randomly and uniformly designate it a specific network to be sent over. Whenever we have a timestep with network i connectivity, we run the $\frac{e}{e-1}$-competitive algorithm of [16]. Since the $\frac{e}{e-1}$-competitive algorithm has the special property that it can operate if only given relative deadlines of the packets, the non-persistance of network connections will not affect its performance.

Theorem 17. *The algorithm above is $\frac{ek}{e-1}$-competitive.*

Proof: Consider the set OPT_i of packets the optimum offline schedule sends over network i. If we are given only this set of packets then we obtain at least $\frac{e-1}{e}$ times the optimum offline value for this network. If we increase the set of packets given, then this can only increase the value we obtain. Now, if packets randomly arrive with probability $\frac{1}{k}$, then in expectation we will be able to obtain at least $\frac{e-1}{ek} OPT_i$. It follows that in expectation $\frac{ek}{e-1} v(ALG_i) \geq OPT_i$. Summing over all i finishes the proof. ■

XII. Acknowledgements

We would like to thank Professor William Kaiser (UCLA), Dr. Maxim Batalin, Zainul Charbiwala and Digvijay Singh for many enlightening discussions about the current state of mobile data transport research.

References

[1] G. Inc. (2010) Gartner says worldwide mobile phone sales to end users grew 8 per cent in fourth quarter 2009; market remained flat in 2009. Press Release. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1306513

[2] R. Entner. (2010) Smartphones to overtake feature phones in u.s. by 2011. [Online]. Available: http://blog.nielsen.com/nielsenwire/consumer/smartphones-to-overtake-feature-phones-in-u-s-by-2011/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed:+NielsenWire+(Nielsen+Wire)

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC '09)*, 2009, pp. 280–293.

[4] H. Falaki and S. Keshav, "Trace-based analysis of wi-fi scanning strategies," *ACM SIGMOBILE Mobile Computing and Communications Review*, pp. 73–76, 2009.

[5] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," in *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys '09)*, 2009, pp. 179–192.

[6] H. Wu, K. Tan, J. Liu, and Y. Zhang, "Footprint: cellular assisted wi-fi ap discovery on mobile phones for energy saving," in *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization (WINTECH '09)*, 2009, pp. 67–76.

[7] J. Wortham. (2009) At&t to urge customers to use less wireless data. [Online]. Available: http://www.nytimes.com/2009/12/10/technology/companies/10iphone.html

[8] ——. (2009) Customers angered as iphones overload at&t. [Online]. Available: http://www.nytimes.com/2009/09/03/technology/companies/03att.html

[9] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[10] A. Kesselman, A. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in qos switches," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC '01)*, 2001, pp. 520–529.

[11] B. Hajek, "On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time," in *Proceedings of the Conference on Information Sciences and Systems*, 2001, pp. 434–439.

[12] M. Chrobak, W. Jawor, J. Sgall, and T. Tichý, "Improved online algorithms for buffer management in qos switches," *ACM Transactions on Algorithms*, vol. 3, no. 4, p. 50, 2007.

[13] F. Li, J. Sethuraman, and C. Stein, "Better online buffer management," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07)*, 2007, pp. 199–208.

[14] M. Englert and M. Westermann, "Considering suppressed packets improves buffer management in qos switches," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07)*, 2007, pp. 209–218.

[15] F. Li, J. Sethuraman, and C. Stein, "An optimal online algorithm for packet scheduling with agreeable deadlines," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '05)*, 2005, pp. 801–802.

[16] Y. Bartal, F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tichý, "Online competitive algorithms for maximizing throughput of unit jobs," *Discrete Algorithms*, vol. 4, no. 2, pp. 255–276, 2006.

[17] A. Coté, A. Meyerson, A. Roytman, M. Shindler, and B. Tagiku, "Energy-efficient online scheduling with deadlines," 2010, unpublished manuscript.

APPENDIX
ONLINE MAXIMUM-WEIGHT MATCHING WITH VERTEX LOCKING

This section provides the 2-competitive algorithm and analysis for Online Maximum-Weight Matching with Vertex Locking described in [17]. To simplify the analysis, we assume that at each time step either one new node of A arrives or one new node of B locks.

The algorithm proceeds as follows: At each time, compute the matching $\mu(A_t, B, F_t)$ which is the maximum-weight matching of the set A_t of A nodes which have arrived by time t with B subject to the requirement that the locked edges F_t be included in the matching (F_t also includes "null" edges for unmatched locked nodes). Whenever a new node locks, add the appropriate edge from our current matching to F_t.

Let $f(A_t, B, F_t)$ be the cost of matching $\mu(A_t, B, F_t)$. Suppose that node $a \in A$ arrives at time t. Define Δ_a to be the change in the cost of the optimum matching due to this arrival. Thus by definition $\Delta_a = f(A_t, B, F_t) - f(A_{t-1}, B, F_{t-1})$ with the observation that $F_t = F_{t-1}$ and $A_t = A_{t-1} \bigcup \{a\}$. For any $b \in B$, define $\rho_t(b) = f(A_t, B, F_t) - f(A_t, B - \{b\}, F_t)$. For any $b \in B$, let v_b be the weight of the matching edge for b in the algorithm's final solution (or $v_b = 0$ if b is unmatched). We observe that the algorithm's total weight is given by:

$$ALG = \sum_{b \in B} v_b = \sum_{a \in A} \Delta_a$$

Lemma 13. *The value of $\rho_t(b)$ is non-decreasing with time.*

Proof: If at time t a new vertex becomes locked then $F_{t-1} \subset F_t$ and $A_{t-1} = A_t$. Since we pick the appropriate edge from $\mu(A_t, B, F_{t-1})$ we have $f(A_t, B, F_t) = f(A_t, B, F_{t-1})$. This new locked edge can only reduce the weight of the best matching to $B - \{b\}$ so $f(A_t, B - \{b\}, F_t) \leq f(A_t, B - \{b\}, F_{t-1})$. Thus, $\rho_t(b) \geq \rho_{t-1}(b)$.

If at time t a new vertex a arrives, then $A_{t-1} \subset A_t$. One way to calculate $f(A_t, B - \{b\}, F_t)$ uses a flow formulation. We add a source vertex incident to each vertex in A_t and a sink vertex adjacent to all vertices in $B - \{b\}$. These new edges will have weight 0 and capacity one. All edges between A_t and $B - \{b\}$ will remain unchanged and have capacity one. Then we can find a maximum weight maximum flow to identify our maximum weight matching (since all capacities are integral, we know our flow precisely defines a matching). We augment B with "dummy" vertices and connect them to the vertices of A with edges of capacity one and weight zero, so as to guarantee that the max-weight flow will saturate all edges from source to A_t. We can also do this to calculate $f(A_{t-1}, B, F_t)$.

Consider taking the two graphs from these two instances and superimposing them (by merging identical vertices). Capacities of any identical edges will add, but we keep their weights constant. Now, all edges are capacity 2 except for those edges incident on a or b. Notice that the individual flows for these two graphs simply add and still form a valid flow. If we consider a maximum weight flow f^* over this new graph, then clearly $weight(f^*) \geq f(A_t, B - \{b\}, F_t) + f(A_{t-1}, B, F_t)$.

We now show that f^* can be decomposed into flows for $f(A_t, B, F_t)$ and $f(A_{t-1}, B - \{b\}, F_t)$ which proves our claim. It is clear that in f^* the unweighted flow value is $2|A_t| - 1$. Thus, each A_t vertex except a has two units of flow passing through it. So starting from a, we can follow the flow, alternating between B and A vertices. This process must stop at a B vertex and so the path has an odd number of edges. We can then add the flow along each odd edge to $f(A_t, B, F_t)$ and the flow along each even edge to $f(A_{t-1}, B - \{b\}, F_t)$. Once this is done, we remove this flow. Note that A vertices now either have 0 units of flow, or 2 units of flow. If b was not involved in this path, then we can start from b and do the same process (being sure to add the flow involving b to $f(A_t, B, F_t)$). Again, the A vertices still have either 0 or 2 units of

flow. Once a and b are handled, then the remainder of the flow can similarly be decomposed. This constructs valid flows for $f(A_t, B, F_t)$ and $f(A_{t-1}, B - \{b\}, F_t)$. Thus $weight(f^*) \leq f(A_t, B, F_t) + f(A_{t-1}, B - \{b\}, F_t)$. ■

Theorem 18. *The algorithm is* 2*-competitive.*

Proof: Consider any nodes $a \in A$ and $b \in B$. Suppose that a arrives at time t, and that ab is a possible match (this requires b not yet locked at time t). We can write the following:

$$\begin{aligned}\Delta_a + \rho_{t-1}(b) &= [f(A_t, B, F_t) - f(A_{t-1}, B, F_t)] \\ &\quad + [f(A_{t-1}, B, F_t) - f(A_{t-1}, B - \{b\}, F_t)] \\ &= f(A_t, B, F_t) - f(A_{t-1}, B - \{b\}, F_t).\end{aligned}$$

The last expression must be at least w_{ab} since one way to form the matching $\mu(A_t, B, F_t)$ involves taking the matching $\mu(A_{t-1}, B - \{b\}, F_t)$ and augmenting by the edge (a, b). If we let τ be the final time for the algorithm (at which we can assume all of B is locked) then by lemma 13 we have $\Delta_a + \rho_\tau(b) \geq w_{ab}$. By definition, we have $v_b = \rho_\tau(b)$. Summing both sides over pairs (a, b) which are matched in the optimum offline solution completes the proof. ■