

Realistic Data Transfer Scheduling with Uncertainty

Upendra Rathnayake^{†,*}, Mohsin Iftikhar^{*,◇}, Maximilian Ott^{*}, Aruna Seneviratne^{†,*}

[†]School of Electrical Eng. & Telecommunications, UNSW, Sydney, Australia

^{*}NICTA, Sydney, Australia

[◇]Department of Computer Science, King Saud University, Riyadh, Saudi Arabia
[first name. second name]@nicta.com.au

ABSTRACT

Next Generation Networks will be comprised of different access technologies. We are already seeing the emergence of mobile devices with the capability of connecting to heterogeneous networks with different capabilities and constraints. In addition, many bandwidth intensive applications have rather relaxed real-time constraints allowing for alternative scheduling mechanisms which can take into account user preferences, network characteristics as well as future network resource availability to better exploit network heterogeneity. The current approaches either simply react to changes, or assume that availability predictions are perfect.

In this paper, we propose a scheduling scheme based on stochastic modeling to account for prediction errors. The scheme optimizes overall user utility gain considering imperfect predictions taken over realistic time intervals while catering for different applications' needs. We use 180 days of real user data of many users to demonstrate that it consistently outperforms other non-stochastic and greedy approaches in typical networking environments.

Keywords

Heterogeneous networking with mobility, Optimization with multi-interfaces, Two-stage stochastic linear program

1. INTRODUCTION

Today, more and more mobile devices are being equipped with multiple network interfaces, and are becoming more powerful both in terms of their processing and storage capabilities. They enable the users to connect to a variety of access networks with different characteristics especially as they move.

The conventional method of improving user utility in these environments is to use vertical handovers [7]. Vertical handovers attempt to provide continuous connectivity as the main objective to support real-time communications. However, many of today's bandwidth intensive applications are non-real-time, such as software updates, podcasts, email and even video on demand (with HTTP streaming) which do not require continuous connectivity. For these applications, it is possible to suspend and resume transfers [23] using network availability predictions and proactive scheduling schemes [1], to maximize user utility.

However, we believe that the techniques proposed to date have not fully exploited the capabilities of the modern devices to cope with the realities associated with the predictions, because they make strong assumptions that predictions are perfect and predictions are possible over arbitrarily longer periods of time. In this paper we propose a practical scheduling model which relaxes the above two unrealistic assumptions and show that using this scheme it is possible to increase utility for users in comparison to the current approaches. In particular, we make the following contributions:

- Describe a practical scheduling model, based on stochastic programming, which operates with non perfect predictions taken over realistic time intervals;
- Demonstrate by using real user data, that it consistently outperforms non-stochastic variants and greedy approaches under numerous conditions; and
- Provide methods to improve the scalability of the scheduling scheme to handle large number of applications and network availability scenarios.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 describes the unrealistic assumptions of the current scheduling models and section 4 provides the details of the practical scheduling model proposed. Section 5 describes the emulation setup with the real user data and section 6 presents results showing how the proposed scheme outperforms other techniques in typical networking environments. An analysis of the computational complexity is presented in section 7 and finally section 8 provides the conclusions.

2. RELATED WORK

It can be convincingly demonstrated that the usage of multiple radio interfaces in a mobile device can dramatically improve the performance and functionality when compared to using a single radio interface [9]. One of the primary focuses of evaluating multi interfaced systems has been exploring the idea of switching among multiple network interfaces to reduce the overall power consumption [8, 10, 11]. In contrast, our work is focusing on improving the overall user utility.

Another focus has been the selection of network interfaces based on policy based mechanisms [12, 13]. Moreover, [18-22] demonstrate how to select the best interface dynamically in a heterogeneous network environment. However, they are merely based on the assumption of using a single network interface at one time. The main motivation behind all of this work was to accommodate continuous connectivity through the interface which will stimulate the needs of applications and user's preferences, while optimizing the vertical handoff decisions. They do not consider tolerance of delay of applications and exploit it to improve user utility as we have proposed.

A method of maximizing user expectations for non-real-time data, by enabling cost and performance-aware selection is described in [14]. The proposed method is static as it uses a pre-specified set of networks for a given application. A method which uses a policy based mechanism for context aware connectivity management is presented in [15]. A scheduling mechanism for transmitting time-critical data in a cost-aware manner over heterogeneous networks has been presented in [16]. Compared to our scheme, none of these schemes anticipate future network/bandwidth availability. Further, a method based on predicting the availability of Bluetooth connectivity has been proposed in [17] to dynamically schedule activities such as data synchronization, minimization of power usage in ad-hoc environments. This scheme too does not consider the simultaneous use of multiple interfaces.

The closest solution for maximizing user utility on multi-interfaced mobile devices in heterogeneous networking environments is presented in [1]. The authors propose a scheduling scheme based on a hill climbing algorithm to maximize user utilities over energy and dollar costs for non-real-time bulk data. They take into account future network availability and use the predicted availability to schedule the data transfers. However, this work assumes that the network availability predictions are perfect and that the predictions can be derived for arbitrarily longer durations, which is clearly not realistic.

Stochastic scheduling, which our proposed scheme is based on, has been extensively studied in multi-processor [24, 25] and multi-server environments [26, 27]. This body of work considers the problem of allocating some resources to requests (processors to programs, servers to customer requests), while addressing the uncertainties associated with requests. However, our proposed scheme differs from these in a number of ways. First, we consider the unavailability of resources; the unavailability of wireless networks. Second, as described in sections 4.2 and 4.3, we look into the resource (network) availability in the future with a "look up duration" and accordingly optimize the utilization of them in the very next time frame ("scheduling duration"). Third, we also consider the cost of resources (the costs of using different networks) which is the key in our idea of gaining more user utility by enabling data transfers over cheaper networks of applications that can tolerate delays. Opportunistic scheduling to optimize cellular network performance has also been well studied in literature [28] but again they consider it from a network operators' perspective whereas we consider it from an individual mobile device's point of view. Moreover, our schedule's granularity is much larger, minutes as opposed to seconds, and we consider resource requests (applications) differently with different utility functions (section 3.3).

3. UNREALISTIC ASSUMPTIONS OF CURRENT SCHEDULING MODELS

In scheduling data transfers for achieving maximum utility and minimum cost, it is vital to have accurate network availability predictions which itself is a challenge. We addressed that in our previous work and proposed several prediction models [2, 3]. With that experience and from others work [29], we make two key observations which need to be addressed in the scheduling, and are described in the first two sub-sections below. Then we explain and elaborate how to select utility functions for applications meaningfully, which are adopted later in the scheduling process.

3.1 Length of Prediction Durations

We observed in [2, 3] that the predictions can not be done for arbitrarily longer periods with good accuracy. Predictions of network availability can be achieved within acceptable accuracy limits when the durations concerned are short (i.e. for 5-10 minutes), but it may not be possible to predict accurately for the whole duration where all the applications' data are transferred (i.e. a period of 15 minutes or more).

This can be addressed by considering only a shorter duration (let us say of 5 minutes) at a time where only parts of applications' data can be transferred. The schedules are then estimated at the beginning of each duration as depicted in

Figure 1. These schedules can be derived in such a way that, the overall utility gain over the cost would be maximized in the long run.

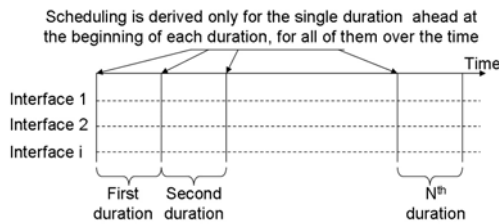


Figure 1. Repeated scheduling over the time.

3.2 Uncertainty in Predictions

As stated in the previous section, it is possible to perform fairly accurate network availability predictions for shorter periods. Still they are not perfect. For example, it is not possible to predict that within the next 5 minutes, Wi-Fi will become available for the first minute, then unavailable for 2 minutes and then again becomes available for the next two minutes. At best one can only provide an indication of percentage availability, e.g. out of next 5 minutes, Wi-Fi will be available for 60% of the time. However at the end of the 5 minutes, we may learn that the Wi-Fi has been available only 40% of the time. This uncertainty needs to be taken into consideration when developing the scheduling model.

This can be addressed by generating the schedule with a stochastic optimization approach. Consider a system which contains 2 interfaces, a Wi-Fi and a 3G where the latter is costlier. Further assume that the interfaces have the same bandwidth and that the duration to be of ten time slots as depicted in Figure 2. Now consider a single application whose utility is a constant which is higher than the costs of both the interfaces and becomes zero exactly after 10 time slots. Further, 5 time slots are necessary and sufficient to complete the data transfer of this application.

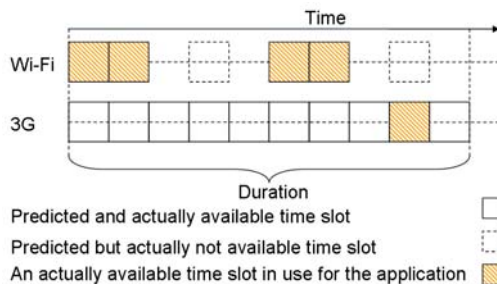


Figure 2. Stochastic Vs deterministic optimization in scheduling.

Now we predict that Wi-Fi will be available 60% of the time (6 time slots) and 3G always (all 10 slots). (The placing of the predicted Wi-Fi time slots over the time line in the figure is arbitrary as we can not predict where it really happens). In reality, Wi-Fi availability is only 40% (4 slots, we explain how to estimate this before the duration ends, in section 5.2) and no difference in 3G (all 10). If we had taken the 60% predicted availability of the Wi-Fi interface as a certain fact and scheduled accordingly to transfer our data, we would have come up with the proposal which suggests only to use the Wi-Fi interface, as 6 time slots will be more than enough to transfer the data. On the contrary, if we had scheduled this as a stochastic optimization problem, we would have come up with a proposal suggesting to use only the Wi-Fi if it is available 60%, but if it's going to be available only 40%, then use that as well as another time slot from the 3G interface for the rest of the data, as a fallback. This will prevent us from gaining a lesser overall utility.

3.3 Selection of Utility Functions for Applications Meaningfully

Utility functions differentiate applications' importance from the user's point of view. Although selecting proper utility functions for applications is not the main focus of this paper, still in this section we try to gain an insight into how to properly select realistic functions which decently represent user preferences.

The utility function specifies the utility of having a data unit of an application at time "t" and these functions assume that even partial download (e.g. web pages, video) or a completion beyond the deadline (e.g. software updates) can be useful, as in [1]. In previous proposed schemes for example, [1] has used linearly decreasing utility functions where the rate of decrease is a constant throughout and the utility approaches to zero after a hard deadline, as illustrated in Figure 3.a. But in

reality, there are two types of applications; real-time and non-real-time. For a large class of non-real-time applications, it is unreasonable to assume a utility function that uniformly decreases over time. For example, if we want to download a software update, it does not matter at what time it is downloaded as long as it is completed within the specified period of time. Therefore we model the utility of this type of applications as a constant up to the deadline. Moreover, it is possible to consider applications having two deadlines; a soft one and a hard one. The soft deadline provides a way to schedule more rigorously, taking risks within that duration as it can even be violated. But the hard deadline reminds us that the data has to be transferred before it. The corresponding utility function is a one that is constant up to the soft deadline and decreases linearly to zero at the hard deadline, as depicted in Figure 3.b. In these cases it is possible to transfer data opportunistically whenever “cheap” networks become available without losing any utility.

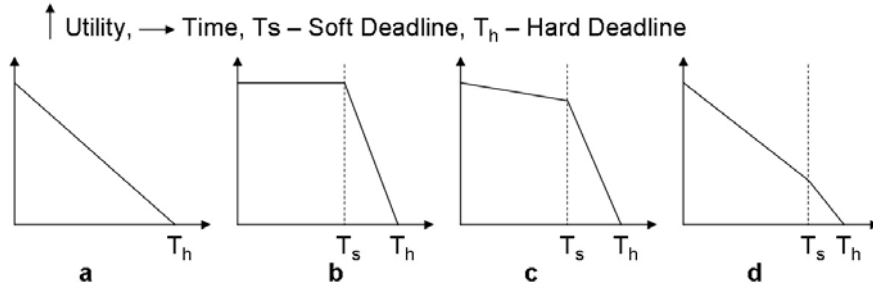


Figure 3. Piecewise linear utility functions.

Even for some (near) real-time applications, if the downloading of the data with a constant rate satisfies the user, the utility function does not have to decrease (at least aggressively) over the time. In the case of a video download for example, if we keep a constant minimum reception rate for smooth playback of the video, then it is unreasonable to assume the utility function to be decreasing fast, as it says in a way that the user is not interested in the latter part of the video file as much as he does at the beginning. We can take a “minimum transfer rate” parameter explicitly with such applications and their utility functions can be represented as in Figure 3.c. where the utility does not decrease aggressively within the soft deadline and after that it approaches to zero at the hard deadline.

For those real-time and other applications where transferring the data “sooner rather than later” really gives higher user satisfaction, we represent their utility functions similar to Figure 3.a, as an aggressively decreasing function even within the soft deadline and approaching to zero at the hard deadline as depicted in Figure 3.d. We call the value of the rate of decrease of an application’s utility function at a time point as its “urgency” at that time. And higher the utility function’s value of an application, the higher its “importance” at that time.

The utility of an application depends on the currently buffered amount of data of that application as well. For example, if the amount of a video that has been already downloaded is enough to play back it for some time ahead, the utility of having more data now may be less. This suggests of having 3 dimensional utility functions where one axis is buffer occupancy (or percentage download); the other axis is time whereas the remaining axis represents the utility. However, we restrict our work in this paper to considering only 2 dimensional piecewise linear utility functions.

4. THE MODELS

First we describe a greedy approach, which is later used for comparison purposes. Then a practical scheduling model based on deterministic optimizations which considers only manageable durations is described. This follows a stochastic addition to it making it a complete model, in section 4.3. In all of our approaches, we do not abandon transferring the data even if the utility of the particular application approaches to zero. We believe that the scheduler should be intelligent enough to consider availability/unavailability of networks and accordingly act to transfer the data cost effectively. In case the networks are not sufficiently available to transfer applications’ data, it may happen that the transfer completions exceed soft/hard deadlines. The effectiveness of the schedulers is measured by taking into consideration the overall utility gains as well as whether the applications’ data were completely transferred within soft/hard deadlines. Furthermore, we adopt the concept of data “bundles” in [1] where applications’ data are sent in small fixed sized amounts called as bundles. A time slot of an interface is the amount of time taken to send such a bundle via that interface.

4.1 Greedy Approach

This approach does not use network availability predictions. It utilizes all available networks to transfer the data. Moreover, it does not consider “minimum transfer rate” of applications but only tries to maximize utility gain. This means, if several applications need transferring data, it first transfers the data of the application whose rate of decrease of utility is the highest, i.e. the most urgent. This is because if less urgent data is transferred ahead; there will be a loss in the overall utility gain as not transferring the more urgent data will result in a larger decrease in the overall utility gain [1]. In reality, if the utility of an application is higher, the more the rate of decrease in general (applications with higher utility happens to be more urgent). Therefore, the alternative policy of sending applications with highest utility first is catered for at the same time to some extent with the above approach. In fact, we used three such applications in our evaluation, as described in section 5.2.

4.2 Shorter Durations: Expected Value Optimization Approach

In section 3.1, we mentioned that it is impossible to predict network availability for arbitrarily longer durations with sufficient accuracy. Therefore we have to schedule data transfers over shorter periods for which, predictions with better accuracy can be obtained repeatedly so that the overall utility gain over interface costs could be maximized. In this approach, we consider this fact and accordingly model it as described below and depicted in Figure 4.

Here we need to find the optimal number of bundles from each application to be sent during the “scheduling duration”, while still satisfying the “minimum transfer rate” (section 3.3) of each application. This will ensure for all applications’ real-time and non-real-time that the user satisfaction is maintained, utility over cost is maximized and each application’s data transfer is completed before the hard deadline.

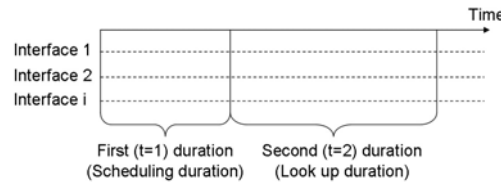


Figure 4. Scheduling for manageable durations.

By considering only the scheduling duration for maximizing utility over cost, if the utility of an application happens to be higher than the costs of at least two of the network interfaces, then data would be sent as much as possible within that duration over those interfaces. But for applications which do not lose utility over the time and can wait, such as software updates, it will be advantageous to look forward and find whether the cheaper networks would become available. To facilitate this, we consider a second “look up duration” in the model as shown in Figure 4.

Let C_j be the cost to send a bundle over an interface j , which is constant for that interface. Further, let $N_{j,t}$ be the number of time slots available to send data over interface j in duration t which will be calculated using the predicted availability of each interface for both the scheduling and lookup durations.

We want to maximize the utility over cost, f . This can be represented by equation (1), where I is the total number of applications, J is the total number of network interfaces, $X_{i,t}$ is the number of data bundles from application i that needs to be sent in duration t , $U_{i,t}$ is the average utility of application i in duration t , and $n_{j,t}$ is the number of time slots used to send data over interface j in duration t .

$$\max f = \sum_{i=1}^I \sum_{t=1}^2 X_{i,t} U_{i,t} - \sum_{j=1}^J C_j \sum_{t=1}^2 n_{j,t} \quad (1)$$

$$\text{Subject to} \quad X_{i,1} \geq \alpha_i \text{ for all } i \quad (2)$$

$$\sum_{t=1}^2 X_{i,t} \geq \beta_i \text{ for all } i \quad (3)$$

$$\sum_{t=1}^2 X_{i,t} \leq \gamma_i \text{ for all } i \quad (4)$$

$$n_{j,t} \leq N_{j,t} \text{ for all } j \text{ and } t \quad (5)$$

$$\sum_{i=1}^I X_{i,t} = \sum_{j=1}^J n_{j,t} \text{ for all } t \quad (6)$$

$$\text{and all } X_{i,t}, n_{j,t} \geq 0 \quad (7)$$

The α_i in equation (2) represents the minimum quantity of data of application i that needs to be transferred in the first duration to satisfy its “minimum transfer rate” as discussed in section 3.3. β_i in (3) represents the same minimum quantity of data that needs to be transferred for both the scheduling and lookup durations and the inequality allows part or all of the minimum amount of data for the second duration to be transferred in the first duration itself. γ_i in (4) on the other hand limits over-using the available interfaces providing suitable applications the ability to wait and typically $\gamma_i = K\beta_i$ where $K(\geq 1)$ is a constant. Equation (6) gives the quantity of data to be sent from each application, which is limited by the number of (predicted) available timeslots given by the equation (5).

What we basically do here is to find out the optimal quantity of data from each application to be sent, and the corresponding optimal number of timeslots to be used from each interface, during both the durations considering the future availability of networks. The order of transmission of applications’ data is not explicit in the model. The transferring is done in the order of application’s urgency at the transfer time within the duration for maximum utility, as explained in section 4.1. Further, only the results for the scheduling duration are taken into account and transferring of the data is done accordingly. The network availability estimates for the look up duration are not as accurate as those for the scheduling duration because it is further ahead the scheduling duration. And also, we can get a prediction for another scheduling duration after passing the current scheduling duration. We use it to run the program for that scheduling duration also (considering another look up duration ahead) and then the data is transferred for that scheduling duration as well. This step by step process is continued as long as we have data to transfer.

4.3 Shorter Durations and Uncertainties: Stochastic Optimization Approach

Despite having shorter prediction intervals, still the predictions for network availability are not perfect. Our own prediction work [2, 3] shows that even with a good prediction model, average prediction error is in the vicinity of 20%, which is significant. For example, we may predict that Wi-Fi will be available 60% of the time, but it might as well be 40% available or 80% in reality, which are called different availability scenarios. We don’t know much about the probability distribution of the predicted availability scenarios which is crucial with most of the stochastic optimization modeling methods.

Two stage stochastic linear programming is a proven method of handling uncertainties in modeling [4, 5]. Furthermore, these models seem simple enough to be implementable in a resource constrained mobile device if the number of variables and constraints considered is less [4, 6]. In its classical form, two stage stochastic linear programming also requires probability distributions of uncertain parameters (i.e. probability distribution of availability scenarios). Despite this, it provides a better leverage for us to observe what the real scenario going to be is and then to adapt to it. Therefore, we use this method in extending our model. With that, we can find the reasonable amount (on average, considering uncertainties) of data from each application to be sent in the scheduling and look up durations, considering the stochastic nature of predictions of both the scheduling and look up durations. This will give some recourse decisions as well, enabling one to later change some of the decisions about the use of different interfaces ($n_{j,t}$ values), depending on the actual availability scenario realized as elaborated using Figure 2 in the section 3.2. This can be achieved by re-estimating the availability in the scheduling duration before the duration ends (in the middle) as we are in a better position there to estimate the availability more accurately than at the beginning. Then accordingly we try to execute the corresponding recourse decision before the duration ends, although we are left only with a half of the scheduling duration to do that.

There is an important observation regarding the data amounts from each application to be sent in the scheduling duration ($X_{i,t}$ values), i.e. these amounts need not be fixed at the beginning of the scheduling duration. That means, normally at the start of the scheduling duration, we decide the quantities of the data to be sent considering uncertainties. Then we re-estimate the real availability at the middle of the duration and accordingly try to send the first decided data amounts at whatever cost depending on the realized availability scenario. But this does not have to be the case. Instead, we can decide the quantities considering the most expected availability scenario of the scheduling duration at the beginning of it and transfer accordingly up to the middle. At the middle of the scheduling duration, we re-estimate and find the real availability scenario and if it is different from the expected scenario, we rerun the program with the actual scenario to find the corresponding quantities, which we could have known at the beginning of the scheduling duration if we had known the

availability in the scheduling duration perfectly. Then we can adapt to these quantities and try to transfer them (although we have only a half of the scheduling duration to do that). Therefore in the model, we explicitly take into account the stochastic nature of the predictions only for the look up duration, but consider only a particular scenario at a time for the scheduling duration. With this flexible approach, we not only increase the overall utility, but also reduce the number of variables and constraints in the model, as we consider only a single availability scenario for the scheduling duration (\bar{w} , explained below) in the model.

Let us think that there are W and Z number of network availability scenarios in scheduling and lookup durations respectively, and a scenario in each duration be denoted by w and z accordingly. Let the probability distribution of z be denoted by $P(z)$. Now assume, $n_{j,1,w}$ denotes the number of time slots used to send data over interface j in duration 1, in availability scenario w ; $n_{j,2,z}$ denotes the number of time slots used to send data over interface j in duration 2, in availability scenario z ; $N_{j,1,w}$ denotes the number of time slots available to send data over interface j in duration 1, in availability scenario w and $N_{j,2,z}$ denotes the number of time slots available to send data over interface j in duration 2, in availability scenario z .

Then the maximization objective function (1) changes as shown in (8), where E is the expectation with respect to $P(z)$, \bar{w} being a particular scenario considered in the scheduling duration and z being any possible availability scenario in the look up duration.

$$\max f = \sum_{i=1}^I \sum_{t=1}^2 X_{i,t} U_{i,t} - E \left[\sum_{j=1}^J C_j [n_{j,1,\bar{w}} + n_{j,2,z}] \right] \quad (8)$$

In the discrete case of $P(z)$, this function becomes

$$f = \sum_{i=1}^I \sum_{t=1}^2 X_{i,t} U_{i,t} - \sum_{z=1}^{z=Z} P(z) \left[\sum_{j=1}^J C_j [n_{j,1,\bar{w}} + n_{j,2,z}] \right] = \sum_{i=1}^I \sum_{t=1}^2 X_{i,t} U_{i,t} - \sum_{j=1}^J C_j [n_{j,1,\bar{w}} + \sum_{z=1}^{z=Z} P(z) n_{j,2,z}] \quad (9)$$

The inequalities (2), (3) and (4) remain unchanged. The equation (5) breaks up to two parts as below.

$$n_{j,1,\bar{w}} \leq N_{j,1,\bar{w}} \text{ for all } j \quad (10)$$

$$n_{j,2,z} \leq N_{j,2,z} \text{ for all } j \text{ and } z \quad (11)$$

And the equality (6) also breaks up to two as

$$\sum_{i=1}^I X_{i,1} = \sum_{j=1}^J n_{j,1,\bar{w}} \quad (12)$$

$$\sum_{i=1}^I X_{i,2} = \sum_{j=1}^J n_{j,2,z} \text{ for all } z \quad (13)$$

$$\text{and all } X_{i,t}, n_{j,1,\bar{w}}, n_{j,2,z} \geq 0 \quad (14)$$

In the objective function f , we use the expected cost with respect to all possible scenarios z , which indeed targets all the possible scenarios and produces a result which is optimal ‘‘on average’’. Further, each $n_{j,2}$ is limited by the corresponding $N_{j,2}$ and the accumulation of $X_{i,2}$ for all the applications should be equal to $n_{j,2}$, for all the scenarios z . Due to the reasons discussed above, we only use a particular scenario \bar{w} of the scheduling duration at a time and therefore it behaves only as a single instance.

5. EMULATION WITH REAL USER DATA

5.1 Real User Data

In our previous work, we have collected around 180 days of real user data of 12 volunteers in Sydney, Australia. Fifteen days worth of data which included Wi-Fi presence and GSM presence recorded for every half a minute were gathered from each volunteer [2,3]. GSM was almost always available for all the users as opposed to Wi-Fi. We predicted the Wi-Fi availability for a 5 minutes duration using all the previous days’ data plus data of the same day just before the prediction time with a learning process, and we did this repeatedly for all the 5 minutes blocks in a day, for the last 5 days of each user. Interested readers are referred to [2] and [3] for more details.

We denote the value, the number of half minutes Wi-Fi is actually available over 10, as the “actual availability” in a 5 minutes duration (found from the corresponding day’s data file). The number of half minutes Wi-Fi is predicted to be available over 10 is the “predicted availability”. Prediction error in that duration is the difference between these two values. We found out that there are higher prediction errors in the mornings and evenings coinciding user travel times and the average was found to be 20% with our best model [3]. In evaluating the schedulers discussed in section 4, we used these travel times with this error figure of 20%. We have predictions for the travel times of five days data (last 5 days of 15 days data) of each user, and this accounts for 60 days’ of real user data of 12 users for the evaluation.

5.2 Emulation Setup

With the above data set together with predictions, we take the scheduling duration to be of 5 minutes, and consider a look up duration of 10 minutes for our emulation. Say we are at a time to schedule our applications data transfers for the next 5 minutes. For that scheduling duration, we already have a prediction from our previous work, we use that as well as the Wi-Fi availability in the 15 minutes just before, to extrapolate the availability for the look up duration (with linear regression). We use the 20% figure as the average error with both of these predicted and estimated availabilities. In executing our schedule, we extract the real availability from the corresponding data file of that particular user to transmit data.

We take a single bundle to be of size 1 KB and consider only the downloading path for the evaluation, although the models can work out for both ways indistinguishably. To emulate a typical data transfer session in mobility, we consider three applications of different types, each having the size of 5 MB, 30 MB and 100MB respectively (we change the sizes later and see the effects in section 6.5). The first one has a higher utility and starts from 70 (in some units, say 1/1000 000 of a dollar as in [1]) and loses it quickly within the first 5 minutes (to 10), then again quickly approaches to zero after the next 2 minutes as in Figure 3.d, resembling an urgent photo transfer for example. The second one starts from 25, linearly decreases to 15 after 15 minutes and then approaches to zero in next 5 minutes as in Figure 3.c, downloading a short video file being a good example for that. The last one resembles a non urgent/non critical application typically like a software update, whose utility starts at 15 and stays there for over 50 minutes, then goes to zero linearly in another 10 minutes as shown in Figure 3.b. Although these utility values are arbitrary, we selected them in such a way that the three applications’ importance is in the decreasing order. Further, we change them to see the effects in section 6.4.

We consider the GSM network presence to be equivalent to 3G network presence, a realistic assumption, and assume that it has 0.5 Mbps fixed rate with cost of 20 in the same units per transferring a bundle (i.e. 2 cents/MB), where the cost can well be a combination of dollar cost, power cost etc. On the other hand, Wi-Fi is assumed to have a fixed rate of 1Mbps with 5 units of cost per bundle (0.5 cents/MB, one fourth of 3G’s). The fixed rate assumption is made due to the fact that we haven’t done the “available bandwidth prediction” in our previous work but only the “availability of networks”, even though our scheduling models are capable of handling even variable rates. Further, we know that 3G and Wi-Fi access bandwidths are diverse, but we consider the above values to be typical end to end rates; nevertheless we change them relative to each other to see the consequences in section 6.3.

In our prediction work, we learned that the actual Wi-Fi availability can be more than the prediction or less than it and also these are equally likely events. Moreover, we found the average prediction error to be 20%. Hence, for the stochastic model (and for the look up duration in the “Perfect” approach described in section 6), we consider only three scenarios of Wi-Fi availability, low, mid and high. The Wi-Fi availability prediction we get is the mid (expected) value, and 20% less or zero percent, whichever the maximum is considered to be the low value and 20% more or 100%, whichever the minimum is considered to be the high value. For example, if the prediction is 90%, the low, mid and high scenarios have availabilities 70%, 90% and 100% respectively. Further, we assume that these low, mid and high scenarios occur with probabilities 0.3, 0.4 and 0.3 respectively (we learned from our prediction work that the low and high scenarios are equally likely and hence the same probability 0.3). In comparison, 3G is always available and therefore, the Wi-Fi availability scenarios become the only scenarios in both scheduling and look up durations. Apart from that, a more accurate $P(z)$ estimation can be obtained by examining the predicted and actual network availabilities over the history of a user. That means, all of the encountered scenarios can first be classified into fewer scenarios and their probability of happening can be obtained by counting their number of occurrences over the time of that particular user. From these counts, the probability of occurrence of a particular scenario can easily be calculated by dividing its count from the total number of occurrences of all of the scenarios.

In the stochastic model, we re-evaluate the Wi-Fi availability of the scheduling duration after passing half of the duration. If the availability in that half is below the low value or, equal and Wi-Fi was not available in the just passed 30 seconds, then the availability of the entire duration is estimated to be low. If that half has more Wi-Fi availability than the high value or, equal and Wi-Fi was available in the just passed 30 seconds, it is estimated to be high. Otherwise, the availability of the

entire duration is considered to be the same mid value. Further, we consider $K = 1.5$ in $\gamma_i = K\beta_i$. All the models were implemented in Matlab using its optimization toolbox.

We start to download these applications simultaneously at the beginning of each travel time. We schedule the transfer of the data of all the three applications together for the immediate 5 minutes and execute it, using all the three methods given in section 4 separately. We reschedule the remaining data for the next 5 minutes, after passage of the current scheduling duration and likewise, we do this process until we finish the transfer of all the data of the three applications. We calculate and accumulate the utility minus cost for all the three applications separately over the time, and also count the time it takes to finish the transfer of the data of each application.

By setting the values for the parameters and for the utility functions of the three applications of the emulation as above, we were able to make the resulting accumulated utility minus cost of each of the three applications with the greedy approach to be equivalent, for all the users on average in the emulation. This ensures that for the total sum, all the three applications contribute equally with the greedy approach, and it may change in other two approaches depending on their performance. We tried our best to set these parameters close to the reality; however it is interesting to see what the repercussions in the results are if we change them. Therefore, we also change each parameter relative to its original value and gain insights into different aspects of these scheduling models in different situations in the latter part of the following section.

6. RESULTS

For evaluations, we use all the three models described in sections 4.1, 4.2 and 4.3 which are identified as “Greedy”, “Expected” and “Stochastic” respectively. Further, we consider the stochastic model with the perfect knowledge of network availability in the scheduling duration (uncertainty in the look up duration remains the same), as to know how far we can go with better predictions, which is identified as “Perfect”. Sometimes the models are unsolvable due to tight constraints, especially with higher values for α_i and β_i in equation 2 and 3. In those cases, we lower the values of these parameters iteratively, in the order of applications’ importance (utility at that time duration).

6.1 With a Typical Data Transfer Session

With the parameter set given in the section 5.2, the utility over cost gained for the three applications were added together to get the total utility. It is shown in the Figure 5, where each value is normalized by the corresponding total utility minus cost with the “Perfect” knowledge approach.

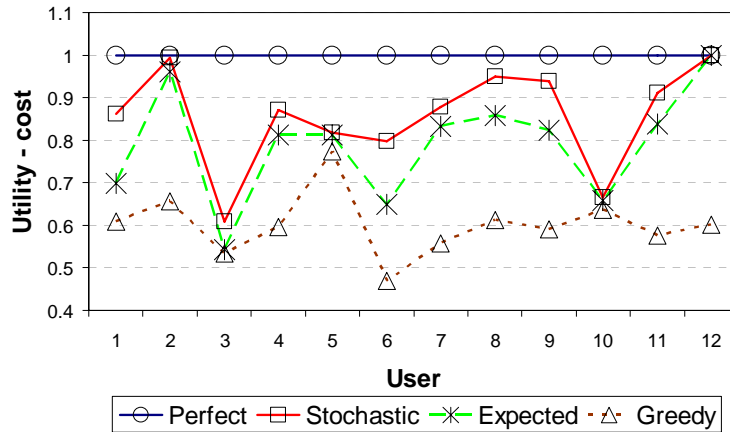


Figure 5. Total utility gain over cost for all applications.

The overall average of all the users for the stochastic model is 0.86, for the expected, it is 0.79 and for the greedy 0.60 and their standard deviations among all of the users are 0.12, 0.13 and 0.07 respectively. We can clearly see that the stochastic model performs far better than the greedy approach (26%), and 7% better than the expected approach on average, showing the adjustability of the stochastic model even with not so perfect predictions. On the other hand, it is only 14% below than if we could have known the future network availability perfectly in the scheduling duration, which demonstrates the

usefulness of our predictions even though the average error is 20%. We show the accumulated utility for each application separately in the Figures 6, 7 and 8.

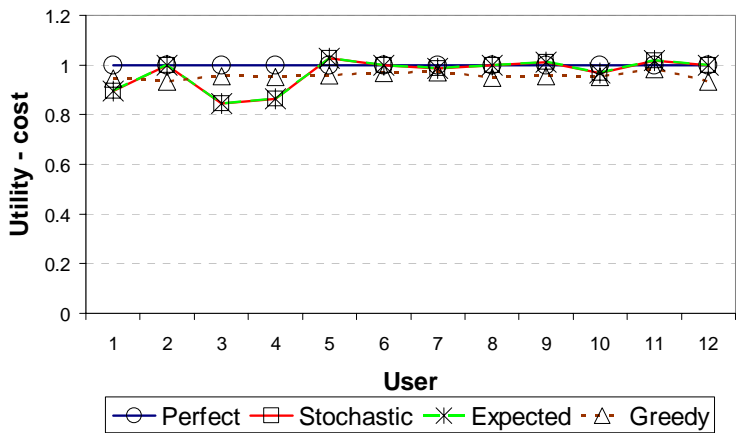


Figure 6. Utility gain over cost for application 1.

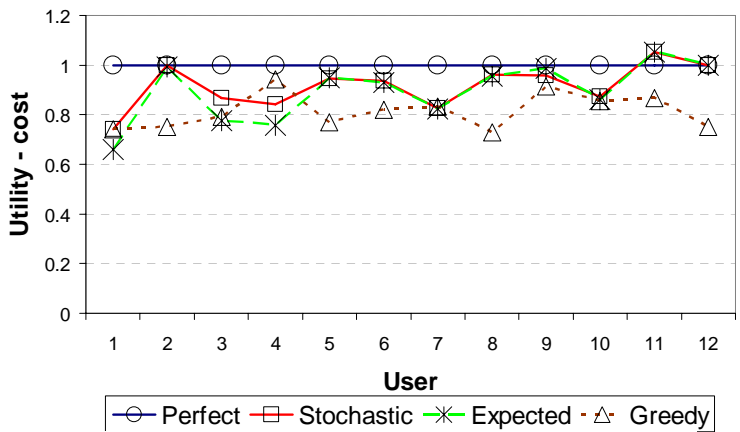


Figure 7. Utility gain over cost for application 2.

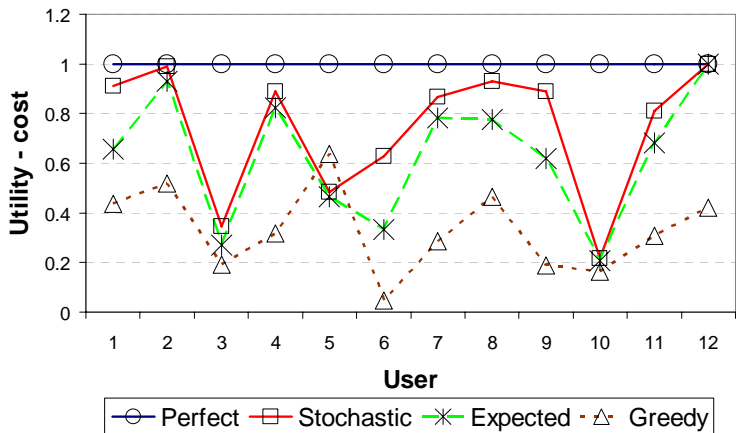


Figure 8. Utility gain over cost for application 3.

These figures clearly indicate that more the applications' capability to wait, the more the utility gain with the stochastic approach than greedy. It is obvious that for more urgent applications, we should use whatever available interfaces, favoring a more greedy approach (Figure 6). But if we consider all the applications together, we can gain more utility with the stochastic approach than the greedy one as it tries to optimize on average for all the applications.

The metric used to compare different approaches is not only the utility gain, but application completion times as well. We show below how each application's data transfer was completed with respect to its soft deadline, in Figures 9, 10 and 11.

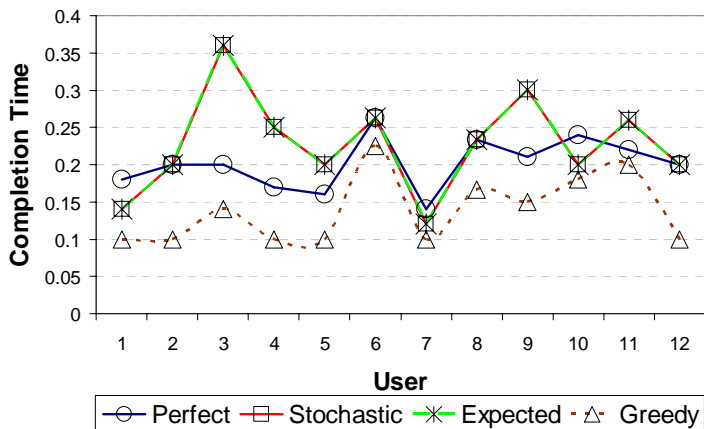


Figure 9. Completion times of application 1.

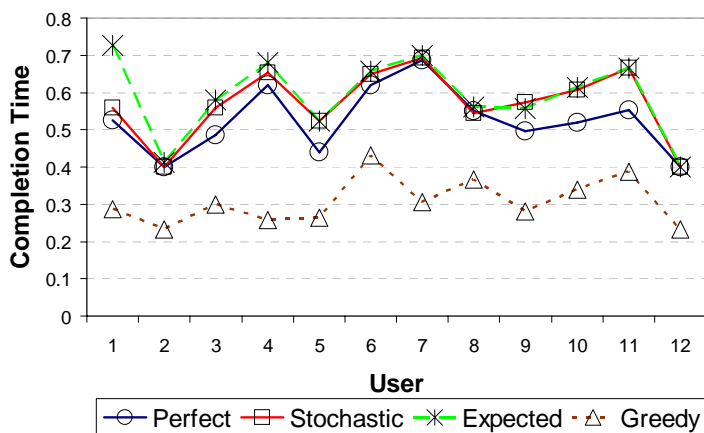


Figure 10. Completion times of application 2.

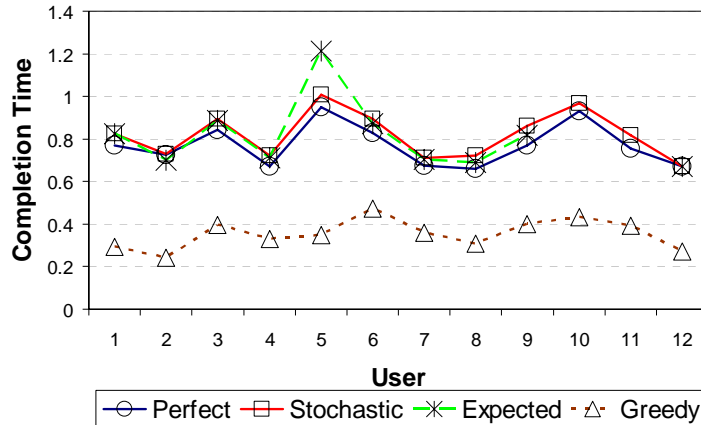


Figure 11. Completion times of application 3.

We can see that the key for gaining more utility is by deferring and rescheduling the applications which can wait! Further, in the long run, “Stochastic” quickly adapts to changes in availability predictions and exploits them to finish more quickly, sparing resources further ahead the time, as seen especially in the Figures 10 and 11. That is very clear in the Figure 11, where for the 10th and 11th users; the application’s data transfer has not finished at all with the “Expected” approach. The reason is, for some travel times of those evaluation days, a Wi-Fi availability of around 0.1 is predicted continuously for scheduling durations towards the end, even though Wi-Fi is actually not available. The “Expected” approach kept on trusting that as a perfect fact and scheduled only to use Wi-Fi whereas the stochastic approach re-evaluated the real availability at the mid of the scheduling duration and if found otherwise to the prediction, executed the corresponding recourse decision, i.e. using 3G in this case. Therefore, we can claim that the stochastic approach is not only better than the expected approach, but necessary for avoiding unnecessary delays even due to predictions with little error. On the other hand, the “Greedy” approach just uses all the available resources irrespective of their costs and finishes the data transfers too earlier than their deadlines, resulting a lesser overall utility gain. But the user is interested in gaining more and more utility and not much in completion times as long as the transfers are finished before the deadlines (that is why for example the utility for the application 3 is constant throughout till the soft deadline).

6.2 With Different Relative Costs

In the above section, we have used 3G cost to be of 20 units and Wi-Fi to be of 5, one fourth of that. We reran the emulation with the same set of parameters but with different Wi-Fi costs, set to 0 and 10, representing zero and two fourth. The results are given below in Figure 12. The values are normalized with respect to the total utility gain with the “Perfect” approach in the one fourth cost case, which is shown in the middle of the figure. The upper and lower sections carry low (zero) and high (two fourth) Wi-Fi cost cases respectively.

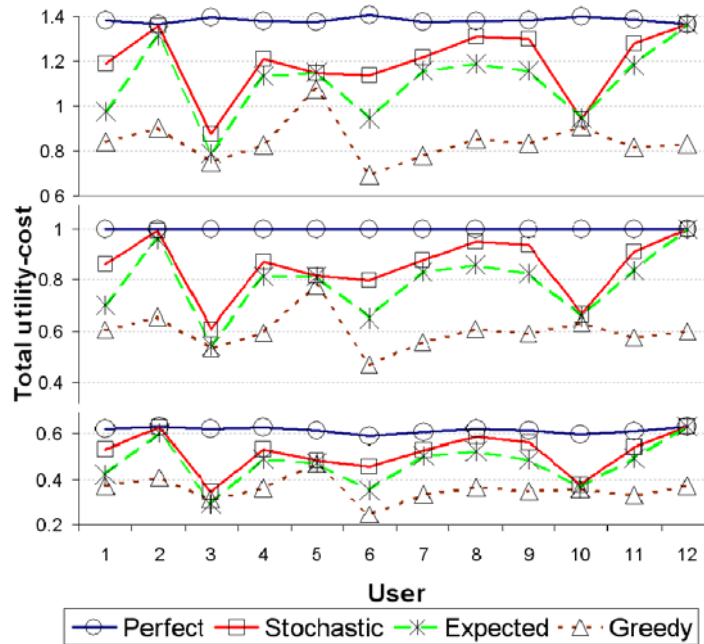


Figure 12. Total utility with different costs.

All the data transfers with all the approaches were completed within the corresponding soft deadline in most cost cases except for some users with “Expected”. As can be seen from the figure also, when the relative cost decreases, the differences between greedy and stochastic approaches widen (in absolute terms), 16%, 26% and 35% respectively in high, mid and low cost cases (but with respect to the utility gain of each case’s “Perfect” approach, they all are around 26%). The same happens between the stochastic and expected approaches, with differences 5%, 7% and 9% respectively. Another observation is that the stochastic approach still performs better than the greedy approach even in the high cost case by 16% (in absolute value terms). We can see that it acts as a practical upper bound for the overall utility over costs and all of them would coincide when the Wi-Fi cost becomes equal to the 3G cost.

6.3 With Different Relative Data Rates

Here we use different rates for the Wi-Fi interface. Earlier we took it to be 1Mbps, two times the 3G rate of 512 Kbps. We ran the emulation with rates 0.5 and 1.5 Mbps for the Wi-Fi, resembling one time and three times of 3G interface’s rate and the results are shown in the Figure 13. Every value was normalized with respect to the value of the “Perfect” approach in the mid rate case. The upper and lower portions of the graph show high and low rate cases, while the mid showing the results with the regular setup.

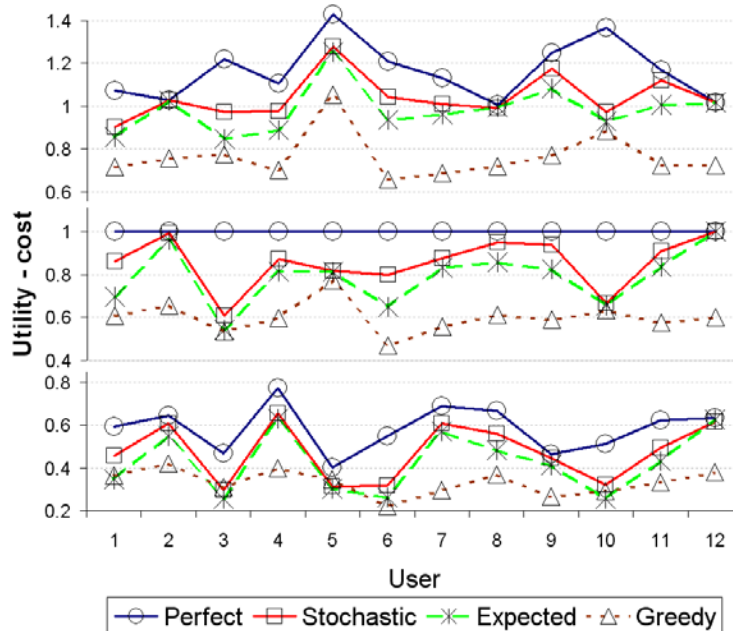


Figure 13. Total utility with different rates.

The difference between greedy and stochastic approaches in low, mid and high rate cases is 14%, 26% and 28% respectively (and with respect to the utility gain of each case's "Perfect" approach, they are 22%, 26% and 24%). In the low rate case, the difference is low understandably as the amount it can send with low cost becomes less. But in the high rate case, it is not as high as we expect which is counter intuitive. The reason behind is that the third application which comparably gains much of the utility with the stochastic approach uses much from the Wi-Fi interface even in the mid rate case, and if the rate becomes high, it only helps to finish the transfers more quickly with an average completion time of 65% (with respect to the soft deadline) compared to 69% in the mid rate case. If the file sizes have been higher, we would have observed a higher difference in the high rate case. Another reason is, when the rate is high, the greedy approach too gains good utility (66% compared to 60% in the mid rate case). Even with different rates, we still see that the stochastic approach acts as a practical upper bound for the overall utility gain over cost.

6.4 With Different Utility Functions (User Preferences)

The parameters we change next are the utility functions. We shift all the utility functions by 10 units up and down with respect to their initial values, which are identified as "high" and "low" cases. The results are given in the Figure 14 where high, mid and low cases are depicted in the upper, middle and lower parts of the figure respectively. Similarly, all the values were normalized by the mid case's utility gain with the "Perfect" approach.

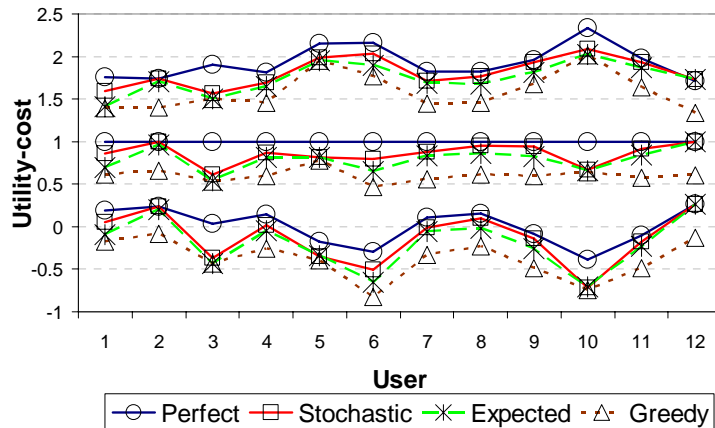


Figure 14. Total utility with different utility functions.

We can see that the utilities over costs have become even negative, which happens because in the low utility case, the utilities of applications are not so important compared to the costs of the interfaces, nevertheless we transfer them. The average difference between “Stochastic” and “Greedy” approaches is 24.88%, 25.67% and 22.89% in low, mid and high utility cases respectively. In the low and mid cases, they seem to be in the same order of magnitude. We can observe that the utility gain has gone comparably down in the high case where the reason would be that the user treats the utilities of all the applications to be comparably higher than the costs and therefore, even the “Greedy” approach produces good utility. That is why the difference between “Greedy” and “Perfect” in high case is only 34%, compared to the 40% in the mid case. This point was further supported by the higher overall utility gain with the greedy approach when we ran the emulation with utility functions shifted further up by 10 units. Overall, we still see the stochastic approach acting as a practical upper bound as in earlier cases.

6.5 With Different File Sizes

To demonstrate the effects of milder and heavier network usage scenarios, we change the file sizes of each application in this section. In the standard (mid) case, they are 5, 30 and 100 MB for the first, second and third applications respectively. We consider two cases; the low case has 2.5, 15 and 50 MB sizes (half of the sizes in the mid case) whereas high case has 7.5, 45 and 150 MB (three halves of the mid case’s sizes). The results are given in the Figure 15 where all the figures were normalized by the mid case’s total utility gain with “Perfect” approach.

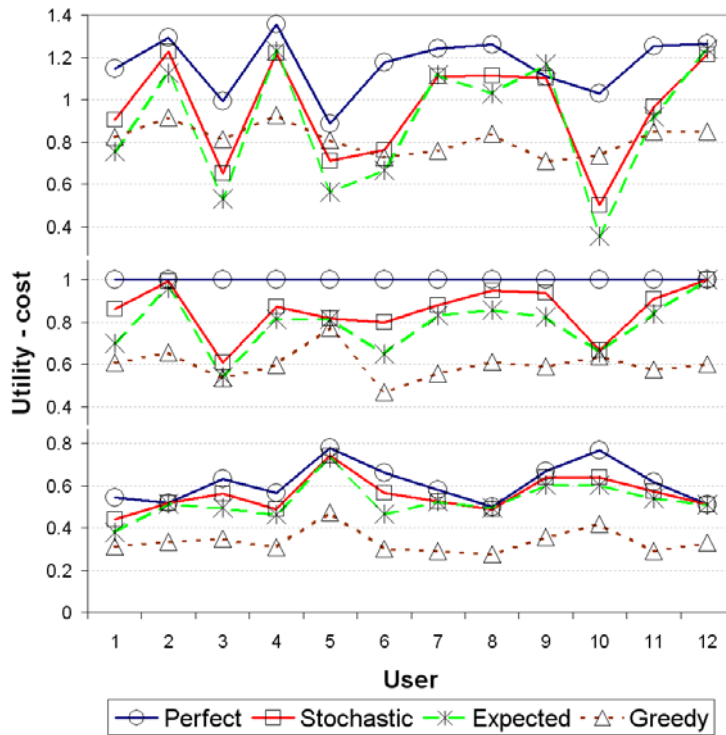


Figure 15. Total utility with different file sizes.

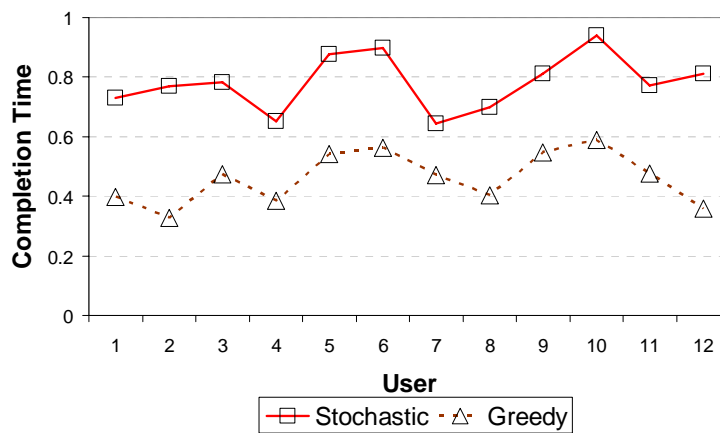


Figure 16. Completion time of the 3rd application with higher file sizes.

The top, mid and lower parts of the graph show high, mid and low cases of file sizes. The difference between stochastic and greedy approaches for them is 14%, 26% and 22% respectively (and with respect to the utility gain of each case's "Perfect" approach, they are 11%, 26% and 36%). The reduction of this difference in the lower case in absolute terms is self explanatory, as when the file sizes decrease, so does the utility gains. But still, with respect to its "Perfect" approach, the difference between "Stochastic" and "Greedy" is 36% showing the effectiveness of the stochastic approach with low file sizes, as it allows more room for optimizations. On the contrary, in the high case the lesser difference as well as greedy approach performing better than stochastic approach in some cases according to the graph is counter intuitive. The clear reason for that is, when the file sizes increase, for some users for whom the network availability is lesser, whatever available interfaces have to be used to transfer them within the deadline, irrespective of the interface costs as there is no much room for an optimization. A greedy approach does exactly the same thing. We can further clarify this point by Figure

16, which shows completion times of the third application (the most vulnerable one due to its size) with these two different approaches (values normalized with respect to the corresponding soft deadline).

We can see a general coincidence that when the completion times with the greedy approach are higher, the overall utility gain over costs with stochastic approach becomes lesser, for example in the case of users 3, 5, 6, 10 and 11. For the user 9, it is higher even though the completion time is still high, the apparent reason is that the predictions for that user have been much better; hence we get an overall utility with the stochastic approach closer to that of the perfect approach. And for the same user, “Expected” results with higher overall utility over costs even than “Perfect”, at the expense of violating even the hard deadline once!

7. COMPUTATIONAL COMPLEXITY

We try to gain an insight into the average computational complexity of both “Expected” and “Stochastic” methods in this section (the “Greedy” approach does not involve any specific problem solving as such). The random variables of the stochastic model, the predictions, were considered to be of discrete nature and with that, the stochastic model actually becomes a specially structured linear program [4], larger in size than the corresponding “Expected” approach’s linear program though. The algorithm we used to solve both the models is the simplex method. Therefore, we compare them in terms of “average case complexity” of the simplex method.

In simplex, if the number of variables involved is r and the number of constraints is s , then the average number of iterations to solve the problem is $0.9*(r+s)$ approximately, including even those cases where simplex encounters degenerate dictionaries along its path to optimality [6]. In our “Expected” approach, $r = I*2 + J*2$, resulting from X_s and n_s and $s = I + I + I + J*2 + 2*2 + I*2 + J*2$ from equations (2) to (7) (note that equation (6) produces two inequalities per one equality). Therefore, the average number of iterations needed for the “Expected” approach is $0.9*(7I + 6J + 4)$.

In the case of “Stochastic” approach, $r = I*2 + J + J*Z$ (X_s , $n_{j,1,w}$ and $n_{j,2,z}$) and $s = I + I + I + J + J*Z + 2 + Z*2 + I*2 + J + J*Z$, from equations (2), (3), (4) and from (10) to (14) (again producing 2 inequalities for each equality in (12) and (13)), and therefore the average number of iterations for a single run of the algorithm is $0.9*(7I + (3+3Z)J + 2Z + 2)$. As described in section 4.3, we run the stochastic algorithm at the beginning of the scheduling duration considering the expected scenario in the scheduling duration and let us assume that the probability of occurrence of the expected scenario is P . At the middle of the scheduling duration, we estimate the availability again and if it is found to be different from the expected scenario (whose probability is $1-P$), we re-run the algorithm which needs the same $0.9*(7I + (3+3Z)J + 2Z + 2)$ average number of iterations. Therefore, the expected average number of iterations is $0.9*(7I + (3+3Z)J + 2Z + 2) + 0.9*(7I + (3+3Z)J + 2Z + 2)*(1-P)$ which results in $0.9*(7I + (3+3Z)J + 2Z + 2)*(2-P)$ total number of iterations where P is between 0 and 1 inclusive. As an example, let us assume that there are 2 interfaces, 3 applications, 3 availability scenarios and $P=0.4$. This necessitates around 75 iterations for the method and if a single iteration needs 1000 CPU cycles, a 500 MHz processor will take 150 micro seconds to calculate the schedules for a single scheduling duration.

In our evaluation, we used 3 scenarios ($Z=3$), 2 interfaces and 3 applications. In reality, the number of interfaces will not be high. The number of applications can increase, but still as in [1], we can always group similar applications together and consider “application groups” in the optimization, as opposed to individual applications, to reduce computations. In Figure 17, we compare the average number of iterations for both the approaches with different number of interfaces (up to 5) and applications/groups (up to 10) with the number of scenarios fixed to 3 and $P=0.4$.

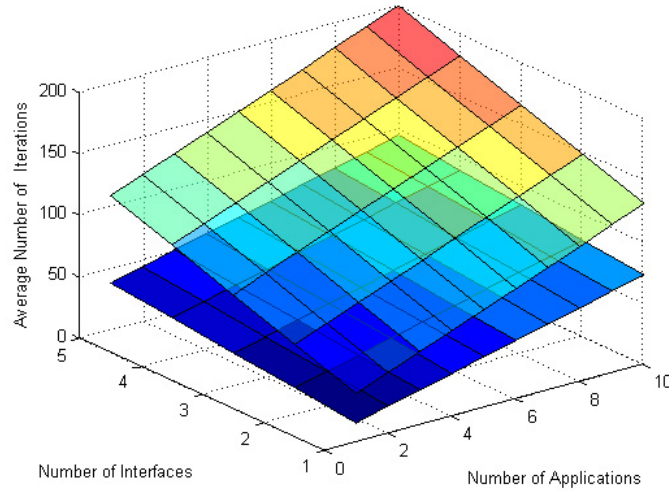


Figure 17. Average number of iterations with stochastic and expected approaches.

The lower and upper planes represent the average number of iterations for the “Expected” and “Stochastic” models respectively. This figure depicts that the average number of iterations for the “Stochastic” model is approximately twice as much as for the “Expected” model. But the number of iterations for the “Stochastic” model will increase more if the number of scenarios Z is increased, which is a critical parameter in number of computations for that model. A solution would be to get some meaningful, minimum number of scenarios with Monte Carlo sampling [5]. Further, in the “Stochastic” approach, we have considered the uncertainties in the “Look up” duration explicitly. This is what adds Z parameter to the number of computations and we can remove it by replacing the stochastic nature from the look up duration with the expected values while uncertainties in the “Scheduling” duration are still being considered (implicitly). This hybrid approach will confine the number of iterations of the stochastic approach to at most twice the same of the corresponding expected approach, making it computationally tractable even for a resource constrained mobile device. We consider this extension as future work.

8. CONCLUSION AND FUTURE WORK

In a heterogeneous network environment, a multi-interfaced mobile device can connect to different access networks with different characteristics at different times due to user mobility. With the prediction of the availability of such networks, it is possible to optimize the usage of access networks for the traffic generated by applications which can bear delays. The main constraints in modeling this problem are that the network availability predictions can not be accurately derived for arbitrarily longer durations and that the predictions can never be perfect. Accordingly, we proposed a realistic scheduling model based on stochastic programming in this paper. We further showed by evaluating with real user data that it outperforms non-stochastic or greedy approaches for different combinations of interface costs, application utility functions and access network bandwidths.

We plan to extend the proposed stochastic model for reducing computational complexity more. Further, even though the simplex algorithm is used as the solver for the stochastic scheduling model in this work, it will be interesting to find out other algorithms for solving it more efficiently with fewer computations, and these solvers may include algorithms such as gradient ascent (hill climbing).

9. ACKNOWLEDGMENTS

This work has been performed in the context of NICTA’s CAMP project, which is funded by Ericsson. We would like to thank the anonymous reviewers for their valuable comments which helped to improve the quality of this work. We would also like to thank to different other people for their insightful discussions in enhancing the work.

10. REFERENCES

- [1] M. Zaharia and S. Keshav, Fast and Optimal Scheduling Over Multiple Network Interfaces, University of Waterloo Technical Report CS-2007-36, October 2007.
- [2] Uendra Rathnayake and Max Ott, Predicting Network Availability Using User Context, in Proceedings of ACM MobiQuitous '08, Dublin, Ireland, July, 2008.
- [3] Uendra Rathnayake, Max Ott and Aruna Seneviratne, A DBN Approach for Network Availability Prediction, In press, MSWiM, Canary Islands, Spain, Oct 2009.
- [4] Julia L. Higle, Stochastic Programming: Optimization When Uncertainty Matters, Tutorials in Operational Research - INFORMS, 2005
- [5] Shapiro A. and Philpott A., A Tutorial on Stochastic Programming, <http://stoprog.org/>
- [6] Vanderbei, R. J., Linear Programming: Foundations and Extensions, third ed., Springer, 2008.
- [7] M. Stenm and R. H. Katz, Vertical handoffs in wireless overlay networks, ACM Mobile Networks and Applications, 3(4): 335-350, Dec. 1998,
- [8] A. Rahmati and L. Zhong, Context-for-Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer, in Proc.MobiSys, June 2007
- [9] P. Bahl, A. Adya, J. Padhye and A. Wolman, Reconsidering Wireless Systems with Multiple Radios, ACM SigComm CCR, vol.34, issue, 5, 10/04
- [10] T. Pering, Y. Agarwal, R. Gupta and R. Want, CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces, in proc. of MobiSys, June 06, Sweden
- [11] E. Shih, P. Bahl, M. Sinclair, Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices, in proc. of ACM MobiCom 2006.
- [12] H. Wang, R. Katz and J. Giese, Policy-Enabled Handoffs across heterogeneous wireless networks, in Mobile Computing Systems and Applications, 2004
- [13] F. Zhu and J. McNair, Optimizations for Vertical Handoff Decision Algorithms, Proc. WCNC 2004
- [14] O. Ormond, J. Murphy and G. Muntean, Utility-based Intelligent Network Selection in Beyond 3G systems, in proc. of IEEE ICC, June 2006.
- [15] Sun J, Riecki J, Sauvola J and Jurmu M, Towards connectivity management adaptability: context awareness in policy representation and end-to-end evaluation algorithm, in proc. of 3rd MUM, college park, MD, 85-92, 2004
- [16] K. Toyama, R. N. Murty, C.A. Thekkath and R. Chandra, Cost-aware networking over heterogeneous data channels, US patent 20070171915, <http://www.freepatentsonline.com/7071915.html>
- [17] Yves Vanrompay, Peter Rigole, Yolande Berbers, Predicting network connectivity for context-aware pervasive systems with localized network availability, in WoSSIoT'07, a workshop of EuroSys, March, 2007
- [18] Bonnin J., Z. B. Hamouda, I. Lassoued, A. Belghith, Middleware for multi-interfaces management through profiles handling, in proc. of Mobileware 2008, Innsbruck, Austria, 2008
- [19] M. K. Sowmia Devia and P. Agarwal, Dynamic interface selection in portable multi-interface terminals, in proc. of IEEE Portable, Orlando, FL, March 25-29, 2007
- [20] A. A. Koutsordoi, E. F. Adamopoulou, K. P. Demestichas, M. E. Theologou, Terminal Management and Intelligent Access Selection in Heterogeneous Environments, MONET 11 (6): 861-871, 2006
- [21] Method and system for selecting an access network in a heterogeneous network environment, US Patent 7315750, <http://www.patentstorm.us/patents/7315750/fulltext.html>
- [22] Schorr, A. Kessler, A. Petrovic G., Adaptive media streaming in heterogeneous wireless networks, in proc. of IEEE Workshop on Multimedia Signal Processing, Sep. 2004
- [23] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, Freeze-tcp: A true end-to-end tcp enhancement mechanism for mobile environments, in IEEE INFOCOM, 2000.

- [24] Wanghong Yuan and Klara Nahrstedt, Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems, in Proc. of 19th ACM Symposium on Operating Systems Principles (SOSP'03), Bolton Landing, NY, October, 2003.
- [25] Changjiu Xian, Yung-Hsiang Lu and Zhiyuan Li, Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time, In proceedings of DAC, Califf, USA 2007
- [26] M. Bayati, M. Squillante and M. Sharma, Optimal scheduling in multi-server queuing network, ACM SIGMETRICS/Performance, 2006
- [27] Jay Sethuraman and Mark Squillante, Optimal stochastic scheduling in multiclass parallel queues, SIGMETRICS 1999
- [28] M. Andrews. A survey of scheduling theory in wireless data networks, Proc. 2005 IMA Summer Workshop on Wireless Communications.
- [29] S. Herborn, H. Petander, and M. Ott. Predictive context aware mobility handling, In International conference on Telecommunications, 2008.