Reconciling Cloud and Mobile Computing Using Activity-Based Predictive Caching

Dawud Gordon, Sven Frauen, and Michael Beigl

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
Firstname.Lastname@kit.edu

Abstract. Cloud computing has greatly increased the utility of mobile devices by allowing processing and data to be offloaded, leaving an interface with higher utility and lower resource consumption on the device. However, mobility leads to loss of connectivity, making these remote resources inaccessible, breaking that utility completely during offline periods. We present a concept for reconciling the fragile connectivity of mobile devices with the distributed nature of cloud computing. We predict periods without connectivity on the mobile devices before they occur and cache process states for applications running on distributed cloud back-ends. The goal is to maintain partial or full utility during offline periods, and thereby to enable an improved user experience. We demonstrate prediction must include real-time behavioral information in addition to location and temporal models. The approach is implemented for mobile phones which learn to quantify human behavior using activity recognition, and then learn patterns in that behavior which lead to disconnectivity. We evaluate it for a streaming music scenario, where data is cached before the user goes offline, allowing seamless playback. The results show that theoretically we can successfully predict 100% of disconnection events on average 8 minutes in advance (std. dev. 46 secs.) with minimal false-positive caching in this scenario, although in the wild these events could prove more difficult to predict.

Keywords: predictive caching, activity recognition, mobile cloud computing, connectivity prediction, mobile apps.

1 Introduction

Smart phones have become pervasive and ubiquitous technology in first and second-world countries. These phones have a tremendous utility in our daily lives, changing the way we conduct many activities [3]. A great deal of this innovation is due to the combination of cloud and server-side computing with local user interfaces on the mobile devices. Cloud computing allows resource consumption to be distributed across many machines. The location of a certain piece of data or execution is not of interest, nor is it known to the user, as long as they maintain connectivity with those instances [9]. This allows the scale of operations to be increased and distributed across many machines, reducing the time for computation for certain types of operations [9]. Pervasive mobile devices

G. Memmi and U. Blanke (Eds.): MobiCASE 2013, LNICST 130, pp. 140–157, 2014.

on the other hand are drastically different in terms of their usage modality. Devices are carried or worn by users, meaning their physical location is of the utmost importance [3]. The devices present an interface to the user, where the interface itself cannot be distributed or remote. Furthermore, since the devices are carried with the user, they are therefore mobile, where mobility inherently implies that loss of connectivity is inevitable [3].

The disparity between these two concepts leads to a conundrum: cloud computing greatly enhances the utility of smart phones when they are connected, but their mobility means at times this connection will be lost. Furthermore, often this increased utility is most needed during times of high mobility, for example using map services while traveling. We present a novel approach for maintaining utility during periods of mobile disconnectivity in order to reconcile these two concepts with each other. In order to solve this problem, resources normally accessible in the cloud must be relocated locally to remain usable when the device is offline. Once the device has lost connectivity, it is already too late to fetch the required resources, therefore the connectivity change must be predicted. Allowing users to explicitly do this themselves is a possibility, but as the number of devices and applications which we use daily grows, this becomes infeasible. The reconciliatory concept we propose is as follows: we look to predict disconnection events before they occur, allowing required resources to be pre-fetched and thereby maintaining at least some utility during offline periods.

The goal is to predict changes in connectivity and detect events which result in insufficient connectivity for applications. Algorithms on the devices monitor the behavior of the human using the device's sensors, generating a time line of quantified human behavior. Within this time line, algorithms then search for and recognize patterns in that behavior which lead to connectivity events. When one of these patterns is identified in real time, the device is alerted before the event occurs, allowing applications running on the device to precache data. Predicting connectivity has been conducted successfully using location [13,15] and time of day [14] as indicators of connectivity patterns. However, for this scenario that information alone does not contain the necessary cues for caching.

Take for example leaving your house every day, where you lose wifi connectivity when you leave. Your location barely changes before you leave the area of connectivity, meaning location-based systems will only be able to react to loss of connectivity, instead of proactively predicting it. While time of day would quite often be a good predictor for leaving, e.g. going to work every day, one often leaves the house at irregular times of day as well. These unscheduled events would not be predictable and would have the same repercussions as not having a prediction-based caching solution for cloud apps. Furthermore, leaving a friend's house would cause the same problems because you don't do that every day, and the location is different than previously recorded [15]. However, identifying someone leaving the house based on observations of their behavior is almost trivial and could be done by any child: putting on your shoes, maybe a jacket as well, leaving the apartment, going down the stairs, etc.. Therefore, we argue that any system which honestly attempts to bridge the gap between cloud

and mobile computing must take all three parameters into account: time, location and human behavior information. We implement a proof-of-concept which runs on a mobile device and attempts to do exactly this using applied machine learning techniques. Since connectivity prediction using location and time of day are large fields and have both been explored independently and in conjunction [13,14], we look at cases in which activity can be used to recognize connectivity state changes which are otherwise unpredictable. We evaluate the approach by observing the performance of the algorithm as it adapts to the behavior of an individual in an experiment described in Sec. 4. The experiment replicates a situation where temporal and location-based connectivity approaches fail, thereby showing the added benefit of using physical behavior as a further indicator.

This evaluation is carried out in two phases and the results are presented in Sec. 5. First we evaluate how well the system performs using the user's own annotations of his behavior as a basis for prediction. We then evaluate the performance of the same system using a timeline of behavioral information inferred from activity sensing data by the device itself.

Using annotated information, the results indicate an overall f-score of only 0.56 for predicting changing connectivity. However, with respect to periods of no connectivity the system performed much better, with a 100% prediction success rate with an average of 8.2 minutes grace period, an f-score of 0.78 and precision of 0.97. Using inferred behavior information caused little change with f-score staying at around 0.58, where the success rate remained at 100% with an average grace period of 8.28 minutes before the event. Based on a behavioral recognition f-score of 0.85 with respect to the annotated labels, precision dropped to 0.79, indicating more false positive predictions, causing damage in terms of unnecessary caching. The final message is that even with imperfect behavior data we can still predict offline periods successfully, but have a higher cost due to unnecessary caching. As behavioral recognition approaches perfection, this overhead approaches 0, allowing systems to seamlessly integrate periods of disconnectivity.

2 Related Work

Sensors integrated into wearable computing systems such as mobile devices can be used to sense human behavior through activity recognition [1,4]. Using activity recognition to quantify human behavior allows devices, systems and applications to detect these activities or contexts in real time and adapt themselves to that behavior, for example by predicting and highlighting the correct app on a mobile device [16]. Using those sensor signals, statements can be made about what that behavior will look like in the future through a process called activity prediction [8]. At the same time, a timeline of recognized and quantified behavior can also be used for predicting future properties, where using symbolic behavioral data in place of numeric sensor data is advantageous in terms of memory and computation for mobile devices [17].

Human beings are creatures of habit [4], making a history of human behavior a good basis for predicting future activities. Using histories of mobile device

connectivity to predict future connectivities however does not yield optimal results due to the fact that "fluctuations of radio quality are too large to make long-term predictions" [5]. There are however other methods for predicting connectivity which serve this purpose better, most focusing on location or time information [15]. Location-based systems work on the basis of connectivity at different locations [13], however, standard behavior at unknown locations causes poor predictions [15]. Research has shown that modeling mobility patterns independent of temporal information (e.g. weekend / weekday) will inevitably lead to sub-optimal connectivity predictions [10]. Using temporal information alone however can also be effective [2], but will inevitably fail if the subject changes their routines or does something out of the ordinary.

Best results can be obtained when combining location with temporal information for predicting connectivity [15]. However even this approach may fail if the subject performs activities at unknown locations, as temporal information only helps to refine location models, or outside of normal temporal routines when connectivity may vary from the normal experience for certain locations [5]. Also, as demonstrated in the introduction, there are still situations in which both time and location are not enough to determine the future state of connectivity.

Resources and execution can be offloaded into the cloud conserving local resources and increasing device utility [6]. However when connectivity is lost, any processes still offloaded are lost and utility suffers. One method for counteracting the loss of utility is to cache resources before going offline. Approaches which use explicit input from the user to cache required resources (data) go back as far as 15 years [11]. More recently automated approaches have been introduced where browsing habits can be used to evaluate and cache links which will probably be clicked [19]. More recently, this concept has been adapted to allow automatic pre-fetching of items with a high probability of being viewed based on a users history and the current network state [7]. The concept we put forth here is to incorporate physical human behavior into temporal and location-based prediction models. We propose a scenario in which both temporal and spatial models would fail, and demonstrate the added benefit of behavioral-based systems.

3 Methods

In order to demonstrate the novel concept for reconciling cloud and mobile computing, we have constructed an archetype for predictive caching. Our system quantifies human behavior using an activity recognition framework designed for embedded recognition. This framework is based on previous work [4] and will not be detailed here. The important aspect for this work is that it quantifies human behavior from motion sensor signals, in this case using supervised machine learning approaches.

In order to preemptively cache resources from the cloud onto a mobile device before we lose connectivity, the device must monitor the following states. The observable state of the sensors embedded in the device S, also known as the evidence, and the connectivity state of the device C. Based on these observations, models can be built by the device from its own experience.

3.1 Behavioral Quantification

The first model required describes human behavior as a function of the sensor signals, allowing the device to recognize and quantify that behavior. This model M consists of models $m \in M$ for each distinct type of behavior $\alpha \in A$ of the form $p(S|a)$, where $s \in S$ are observations from the sensors of the device. Using this model, a prior distribution, and given a set of observations s, the device can calculate a probability distribution over these models, or $p(a|s)$ using Bayesian inference and the law of total probability [12]. For each discrete type of human behavior α, we model the belief function over the evidence, or extracted sensor features, as a probability density function (PDF):

$$m_\alpha = P(S|\alpha) = \sum_{k=1}^{K} \pi_k \mathcal{N}(S|\mu_k, \Sigma_k)$$

We then use these PDFs to generate a posterior probability distribution across behavioral states using the law of total probability and a prior distribution:

$$p(\alpha|S) = \frac{\frac{P(S|\alpha)p(S)}{p(\alpha)}}{\sum_{\alpha_i}^{A} p(\alpha_i|S)}$$

Using this posterior, we can then recognize human behavior by selecting the most probable model at any given time as the behavior for that time slice:

$$\arg\max_{\alpha} p(\alpha|S)$$

This inference, or activity classification, is conducted periodically, generating a time line of human behavior $\boldsymbol{A}$.

3.2 Connectivity Prediction

Based on the observations of the network connectivity states $c \in C$, transitions $\phi \in \boldsymbol{\Phi}$ between network states can also be observed. By observing $\boldsymbol{A}$ and $\boldsymbol{\Phi}$ in parallel, $\boldsymbol{A}$ can be mined for behavioral patterns which occur directly before specific transitions in $\boldsymbol{\Phi}$. Due to the temporal relationship between the two, a pattern in $\boldsymbol{A}$ which is an antecedent to specific ϕ can be observed as having a causal relationship with that transition. For a given pattern length t_p, we model the causal behavioral patterns for transitions with the same resulting state as a Markov chain. Specifically, we model segments of behavior of length t_p leading up to those transitions.

In other words, the periods of behavior leading up to a change in network connectivity state into a specific state are modeled together in order to be able to later predict a transition into that state. This set of models is then used

for predicting future changes in network state. Because we also model the null transition, we can use the law of total probability again to generate a distribution of probability across the models. The model with the highest probability is then output as the predicted connectivity transition.

Each ϕ_{ij} represents a switch from network state c_i to state c_j. The model is then built by observing a history of transitions $\boldsymbol{\Phi}$ between these states, and calculating the transitional probabilities as a Markov process. The result is a Markov model of transitional probabilities between all connectivity states of the mobile device. For each transition ϕ_{xi} *into* a new connectivity state c_i regardless of the previous state c_x, the activity time line of length t_p leading up to the transitions $\boldsymbol{A}_{t-t_p,t}$ are modeled together in Markov model $\omega_{c_i} \in \Omega$.

These activities are modeled as a Markov chain, where the model for transition into connectivity state c_i takes the following form: $\omega_{c_i} = (A, \lambda)$. where A are the states, namely one each quantization of the human behavior (recognized activity) extracted previously. λ are the transitional probabilities between activities, modeled on the data $\boldsymbol{A}_{t-t_p,t}$ leading up to the connectivity transitions, or:

$$\lambda : A \times A \to A, \text{ where } \lambda_{ij} = p(\alpha_t = \alpha_i | \alpha_{t-1} = \alpha_j)$$

For each connectivity state c_i we now have a Markov chain modeling the human behavior which "causes" transitions into that connectivity state, ω_{c_i}. For a given timeline of behavior $\boldsymbol{A}$ of length t_p, we now need to be able to calculate the probability that this history will lead to a transition, and if so which one. To do this, all Markov chains are traversed in parallel by taking the product of the transition required given $\boldsymbol{A}$ as evidence. Using the law of total probability, the probabilities are normalized across all models, and the most probable model is output as the prediction of the future connectivity state for the device:

$$\text{Prediction} = \arg\max_{c} \frac{p(\omega_c|\boldsymbol{A})}{\sum_{c_i}^{C} p(\omega_{c_i}|\boldsymbol{A})}$$

3.3 Predictive Caching

The combined approaches for recognition and prediction are shown in Fig. 1. Once the framework has predicted the future network connectivity state, this information is provided to every software entity on the device which is interested. At this point each application must then form its own decision about the appropriate action to take. The first input into this decision making process is the level of connectivity required in order to deliver the user experience which is wished. For example, for certain applications such as background data synchronization tasks, low speed and bandwidth may be acceptable, while for others such as streaming video applications, even slight reductions in connectivity may be unacceptable. Once the threshold in connectivity for the app has been predicted, the app can then decide to prepare itself for an offline period by prefetching resources from the cloud.

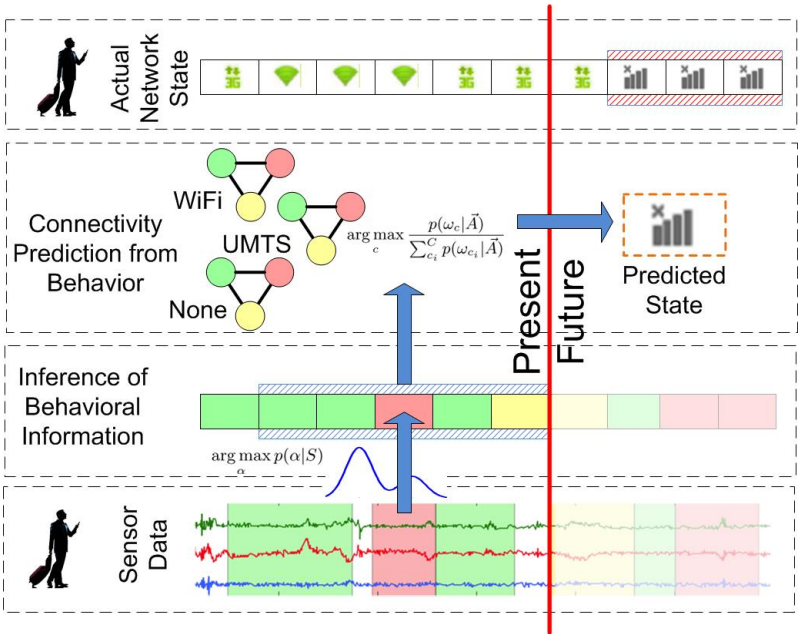


Fig. 1. The architecture of the proposed concept

The next decision is what to prefetch and cache. This decision is also based on what is required for optimal performance, and what is predicted. For example, using mobile map services with a slow connection may be enough to perform online searches comfortably if the map data has been cached. For the scenario evaluated here, as soon as the connectivity drops below the minimum for streaming music, the user experience plummets and the data must therefore be cached. Not only the type of data must be selected, but also which instance, e.g. which song or songs, or which map area. Each app using the framework is best equipped to understand what the usage profile of the user for that app is, and therefore which resource is required to restore the local process' state and preserve the experience as best as possible. How much of the experience and the utility can be preserved is dependent on the type of app, the user and the scenario, and is therefore not evaluated. We examine how well the framework provides apps with information about future connectivity, and the correctness of that information.

3.4 Evaluation Metrics

In order to evaluate the system we need to first define what we want from it, and then create metrics for assessing that [18]. What we want from the system is to inform us of the connectivity changes in the future based on a pattern of human behavior. What we know is the length of time over which these patterns

occur, namely t_p. Based on this information we can define metrics to evaluate the system.

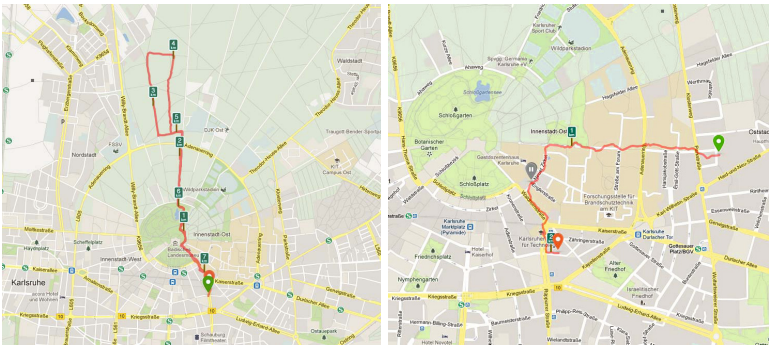
We define a **true-positive (TP)** prediction for a connectivity change to state c_i as the existence of transition $\exists_x | \tau_{xi} \in t_p$ within the time frame t_p of prediction. If the predicted transition to state c_i does not appear within t_p, or $\nexists_x | \tau_{xi} \in t_p$, this is then classified as a **false-negative (FN)** prediction. A prediction of the null transfer τ_{nc} when no transfer occurs $\forall_{x,y} | \tau_{xy} \notin t_p$, or the prediction of a transfer to the current state of the system, are both counted as a **true-negative (TN)**. If the predictor predicts any transition $\tau \neq \tau_{nc}$ and no transition occurs, or $\nexists_{x,y} | \tau_{xy} \neq \tau_{nc}, \tau_{xy} \in t_p$, this is **false-positive (FP)** (i.e. FP of 'no connection' results in unnecessary caching). Using these values we can calculate accuracy, precision, recall and f-score [12].

When evaluating the performance of prediction for transitions to a single class alone, the evaluation is slightly adapted. The metric is applied only to instances of that class alone, where all other classes are grouped together in a single class, as a two class problem. The reason for this is that transition to and between other states should not affect the results for a single class evaluation, which is not the case without this exception. Transitions into other classes within t_p are no longer counted as false-positives, as when observing only that class and the null class, these are then true-negatives.

These metrics allow us to evaluate the overall system performance. However they do not capture one critical aspect of system behavior. Take the example of retrieving something from the basement of an apartment building. In the apartment we have WiFi and in the stairwell we have a mobile data connection. In the basement for a short period we have no connection, followed by mobile connectivity on the stairs and WiFi again in the apartment.

In this scenario, if the behavioral pattern window t_p is larger than the length of the offline window, then completely missing the prediction of the connectivity loss would still result in an f-score of 1 as long as the other transitions inside the window are correctly predicted. In order to account for this, a different metric is required, which takes this into account. We define the prediction **success rate** as the number of transitions to that state which are predicted inside of t_p before they occur, divided by the total number of network transitions. Note that as with the other metrics, success rate can also be applied to a single transition type.

In addition to whether or not a specific transfer is successfully predicted, it is also important to note how far in advance the prediction came. Here we introduce the **time-to-event** metric to evaluate how long the system has to react to prediction information. The metric is defined as for all successful predictions, the time between the window and the occurrence of the connectivity transition event. It is important to note that as a result of the definition of success rate, this value will always be less than t_p. This information is especially critical for predictive process state caching, as it indicates whether the system is actually able to cache the required information in time before the connectivity event.



(a) Jogging route (b) Commuting route

Fig. 2. The jogging route and the commute path taken in the data set

4 Experiment

In this work we propose to reconcile the concept of cloud computing with the fragile connectivity of mobile devices. To evaluate behavior-based precaching as a way to address these differences, we conducted an experiment in a specific usage scenario with a cloud-based streaming music service.

4.1 Scenario

During a normal jogging route, the subject jogs through an open park where there is little or no connectivity. He prefers to use a streaming music service which tailors a playlist for him based on his preferences and those of his friends. However, when he is in the park the slow connection is not sufficient to stream music. He is a student, and therefore does this at different times of day, meaning it is difficult to predict the event of jogging using temporal models. The beginning of his jogging route is along the same path as his trip to campus, making it difficult to differentiate the two using location models alone (see Fig. 2). Even the speed at which he travels does not differentiate between going to campus and going for a run, as he sometimes walks or rides his bike depending on his mood and the weather. However, when looking at the physical behavior, there are subtle differences between going to work and going jogging.

The challenge is that by the time the user enters the park, it is already too late to begin precaching, meaning this information must be present beforehand. At the time when the signal for precaching is required, both location and temporal features for going to work and going jogging are indiscernible from each other. In this case however, an intelligent version of the music service would not have a problem deciding what needs to be cached (the process state) in order to continue execution offline: the play-list.

Fig. 3. Left: mobile phone running the novel framework and the SmartWatch showing prediction state (color) and behavioral history. Right: the view for labeling activities and controlling the framework state.

4.2 Framework

For this experiment an HTC Desire Bravo was used as a mobile sensing device. The device contains a light and temperature sensor for sensing the environment, as well as an acceleration and orientation sensor for sensing physical behavior. The device was carried in the users pocket with the same orientation throughout the entire experiment. Paired with the device was a SONY SmartWatch[1] which was used to visualize the state of the mobile device without disturbing it for the purpose of activity recognition. The SmartWatch was also used as an input device for the predictive framework. The application for the SmartWatch and connected Android device can be seen in Fig. 3, where both the system output and input mode can be seen. This device allows the subject to interact with the monitoring device without having to physically touch it, thereby disturbing the behavior monitoring as little as possible.

The prediction of future connectivity was realized with a software framework running entirely on the device. The framework is implemented as a service for Android which runs in the background. All sensors are sampled at a dynamic sampling rate which is controlled by the operating system, with real sample rates averaged around 30 Hz for the behavior sensors.

Signal features are extracted over 2 second windows of sensor data, with a 50% overlap between windows, outputting a feature vector every second. The features calculated by the system are signal mean, median, standard deviation, min-max difference, signal entropy, FFT frequency peak and FFT entropy. These are calculated for each axis of the acceleration and orientation sensors individually as well. In practice only a subset of these features is required to achieve the same accuracy or negligible reductions, reducing computational load on the device. The necessary features were identified using standard feature selection algorithms [12].

Each second these features are classified into 6 different motion classes, standing, sitting, walking, climbing stairs, running and 'other'. This is done using a

[1] `http://www.sonymobile.com/us/products/accessories/smartwatch/`

probabilistic classifier which models the data for each activity as a mixture of Gaussians, where the number of components is estimated by the system using expectation maximization [12]. For this purpose, naive Bayes classifier was adapted to be trained and to classify on the Android device. The framework is based on previous systems which run on a mobile device and carry out both recognition and prediction actions simultaneously [4].

4.3 Data Set

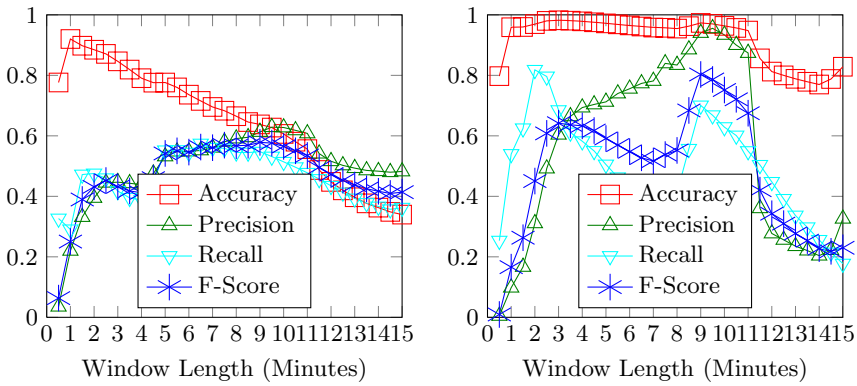
The data set collected here is from one individual and was collected of the course of several weeks. The subject is male, 26 years old, 187 cm tall, weighs 87 kg, and is an Information Management student. The concept here is not to deploy a single system which works for all subjects (generalized), but to research an approach which allows devices to learn and adapt to their users over time (personalized). We argue that to demonstrate this concept, a longer term data set from a single subject is more suitable than having several shorter-term instances from multiple subjects. The data set contains sensor data from the acceleration, magnetic field, and GPS (not used for recognition) sensors on the device. Activities performed were labeled by the subject himself, using a touch-screen interface on the SmartWatch designed for this purpose. Network connectivity was also recorded and is annotated in the data set. Each trip is stored in a separate file where the type of trip, either jogging or commuting to or from campus, is indicated in the file name.

From this data we selected 1450 minutes of sensor data to use for this experiment. This contains 10 jogging runs of approximately 65 minutes each, including a period of time before departing from, and after arriving at the apartment. The daily trip to campus and back was also recorded, in total 10 round-trip commutes, approximately 20 minutes each way. During jogging, loss of connection was simulated using a geo-fencing approach. Crossing the boundary of the wooded area caused the mobile phone to lose network connectivity, motivating the scenario. In practice the system encountered and quantified the connectivity states of WiFi, UMTS, EDGE and NONE, but as EDGE did not occur at all in the traces, and because it is also not sufficient for streaming music purposes, this state was excluded from the evaluation and used as NONE whenever encountered. The daily commute begins with a similar pattern as going for a run, but does not lose connectivity. This data set has been made public as part of the contribution of this work[2].

5 Evaluation

In this section the performance of the predictive caching algorithms during the course of the study will be evaluated. The data presented here is generated using a leave-one-out evaluation approach. For each different type of trip (either

[2] `www.teco.kit.edu/~gordon/precaching/data_set.zip`



(a) All connectivity states (b) 'No connection'

Fig. 4. Results over t_p using behavior labels

jogging, going to campus, or going home), one trip is set aside and the others are used for learning with respect to activity recognition and the behavioral pattern for causal transition modeling. The performance is then evaluated using the data that was set aside. This process is repeated until each trip instance has been used for evaluation once and the results are averaged across all iterations.

The evaluation is two fold. First we evaluate the ability of the reference implementation to predict future connectivity. This is done using behavioral annotations provided manually for the prediction process, representing the minimum of error possible for behavioral information: the ability of a human to discern these behaviors. Although these annotations contain only human error, a system which attempts to recognize behavior automatically will introduce further error into the behavior annotations. At best, the system will learn to decipher these behaviors, but will never be able to perform better than the human, as it uses the human labels for training, and is evaluated by how well it is able to fit them. For this reason, the second part of the evaluation is concerned with investigating the performance of the same system using the error-bound output of a behavioral recognition system for prediction.

5.1 Proof-of-Concept

In this section we evaluate if behavior is indeed an indicator of future connectivity. The basis for prediction in this section is a time line of activity labels as annotated by the user through the SmartWatch interface. In the next section the performance when using error-bound activity data will be evaluated. In order to assess which behavioral patterns best allow us to predict connectivity changes, we dynamically changed the length of time before the change in connectivity state which was modeled t_p. This parameter was varied from 30 seconds

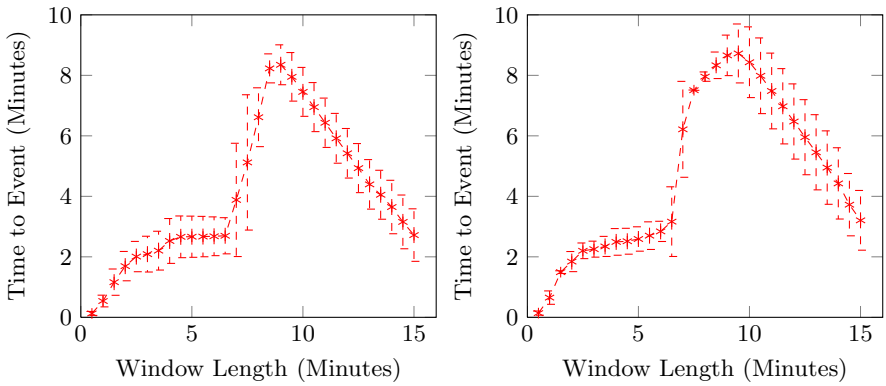
to 15 minutes and the accuracy, precision, recall and f-score were evaluated. The results of this evaluation can be seen in Fig. 4(a).

The first thing to note is that accuracy is high, even for small window sizes. Precision and recall are however quite poor, leading to a poor f-score as well. This is caused by the imbalance between the few number of transitions from one connectivity state to another, and the large amount of data containing no transition. For small window sizes, a high number of steps where no transition occurs in the near future are correctly predicted as the 'no change' class. However, the low f-score indicates that when transitions do occur in the near future, these are misclassified. Accuracy is however known to be susceptible to non independently and identically distributed (i.i.d.) data [18], and is therefore not of much interest for this evaluation other than for indicating the correctness of the 'no-change' prediction.

As the window size increases so too do precision, recall and f-score, although not monotonically. The periodicity would seem to indicate that there are patterns of different lengths which are tell-tales of connectivity transitions, or possibly different harmonics of the same pattern. As the number of 'no-change' windows is reduced (longer patterns mean predictions extend longer into the future), the accuracy falls. F-score reaches an optimum of around 0.56 at 9 minutes, indicating the length for the most decisive patterns for predicting transitions in this scenario. Nonetheless, the initial values here appear to be a negative result, as an f-score of 0.56 would seem to indicate the inability of the system to allow prefetching of distributed resources.

This would appear to be confirmed by Fig. 4(b) which shows the results only for the specific transition to 'no connection'. Again accuracy is not an indicator of actual performance as the number of negative instances far outweighs the number of positive instances. The f-score however far exceeds overall values in Fig. 4(a), with an optimum at around 0.78 at a window length of 9 minutes. Interestingly, the precision for the unconnected state prediction is exceptionally high at that point as well at 0.93. This indicates that only in exceedingly few instances would an app cache resources unnecessarily. The next step is to evaluate how many instances of no connectivity were actually predicted ahead of time (success rate), and how far ahead of time (time-to-event). We then evaluate the success rate over the same window sizes to see what the user would experience from system performance. Despite the poor results indicated by the the recall metric which shows only 0.67 in Fig. 4(b) for a window length of 9 minutes, the success rate is 100% with all offline periods being successfully predicted. So how is a perfect prediction success rate possible with such low recall?

The low recall rates come about when many instances which should be predicted as 'no connection' are predicted as something else. The cause of these low recall rates is that after having correctly predicted the loss of connectivity in the future, the pattern for maintaining the loss of connectivity (the 'no change' model) becomes more dominant than the pattern for losing connectivity. However, while the transfer from a connectivity state to no connectivity is in the near future, this is judged as a missed prediction. High precision however indicates



(a) Behavior labels (b) Inferred behavior

Fig. 5. Time-to-event over t_p using behavior labels (a) and inferred behavior (b).

that the system is not generating false positive predictions, meaning that our streaming application is not caching when it doesn't have to. While this would not directly affect the user's experience of the system with respect to listening to music, it would have repercussions with respect to resource consumption. Most importantly, caching when not necessary causes battery consumption due to processing, storage and wireless communication, which will at some point negatively affect user experience.

Now the next question is, did the predicted warning arrive in time to allow the device to cache the playlist? In Fig. 5(a) the time-to-event for predicted loss of connectivity is shown across window length. Here we can see clearly that for the optimal behavioral pattern window of 9 minutes, a mean time-to-event of 8.20 minutes with a standard deviation of 46 seconds are obtained. Before the offline periods the device uses mobile internet. The measured connection at the location during caching is 1000 KBit/s, or 125 KByte/s, meaning 60 MB. Assuming 3.5 MB and 3 minutes play time per song, that equates to around 17 songs, or 51 minutes of music. The offline periods themselves last for 9.4 minutes on average, meaning the user would even have the luxury of being able to skip 13 songs they don't like.

5.2 Error-Bound Behavior Data

In the section we evaluated the performance of the system when using error-bound behavior data as recognized by the system using the sensors. The recognition system is based on a previously published embedded classification system for android phones [4] which has been simplified to the level described. During the leave-one-out evaluation, the training data is used to model classifiers for recognizing the behavior in real time from device sensors. The results of the 10-fold cross-validation are shown in Tab. 1. Here the classification errors can be seen, representing error in the activity time line. The accuracy of the system

is 0.97, but again this is not representative of system behavior due to non-i.i.d. data [18]. In reality the behavioral recognition system achieves a recall of 0.91 and precision of 0.80, yielding an f-score of 0.85. These values are realistic for many activity recognition systems, and better rates are often achieved in the literature [1,4], making this a fair evaluation of the system.

Table 1. Confusion matrix in percent for the activity recognition chain

a Sit	**b** Stand	**c** Walk	**d** Stairs	**e** Jog	**f** Other	
99.2	0.0	0.0	0.0	0.0	0.7	**a**
0.0	92.5	1.7	3.9	0.0	1.9	**b**
0.0	2.1	87.9	9.6	0.2	0.4	**c**
0.0	0.9	7.1	91.1	0.2	0.8	**d**
0.0	0.1	0.3	0.1	99.4	0.1	**e**
0.0	14.2	0.8	11.1	0.0	73.9	**f**

When observing overall system performance, using recognized behavior for prediction does not appear to have drastic consequences for the systems. This can be seen when comparing Fig. 6(a), containing the results using recognized behavior, with Fig. 4(a) which was achieved using labeled behavior. The maximum f-score of the one is 0.56 compare to 0.58, both at window lengths for pattern modeling of 9 minutes. This would seem to indicate that although the data is to some extent flawed, the predictor trained on a timeline containing these flaws is able to nonetheless predict connectivity correctly.

However, when observing the prediction of the 'no connection' state, we can see that this is not entirely the case. Fig. 6(b) contains the results for prediction

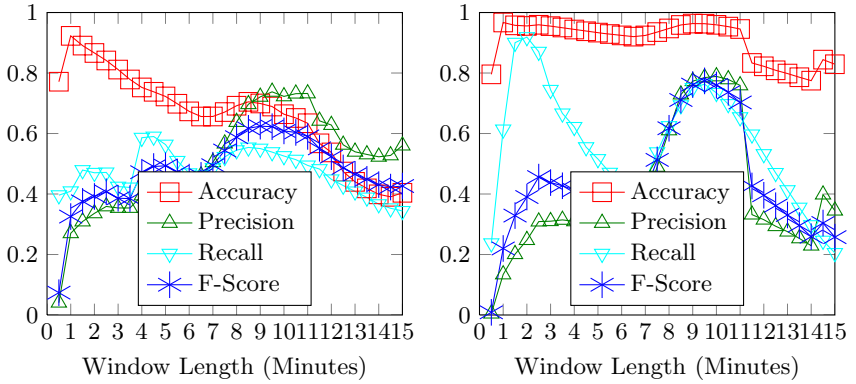


Fig. 6. Results over t_p using inferred behavior data

of these state transitions alone. When comparing the same results using labeled data in Fig. 4(b), we see a drop in maximum f-score from 0.78 to 0.71, and a far more drastic a drop in precision from 0.93 to 0.79. This indicates that using error-prone behavior data for prediction causes a higher false-positive rate, causing the system to cache unnecessarily. Again though, across the board and also at a window length of 9 minutes, the system achieved a success rate of 100% even using error-bound behavioral data.

Finally, we looked at time-to-event occurrence for inferred behavioral data, the results of which are shown in Fig. 5(b). Here for window lengths of 9 minutes, 8.28 minutes with a standard deviation of 49 seconds was achieved between predictions and the occurrence of the offline event. This is a few seconds more than the same value using labeled data, and is also sufficient for the scenario.

6 Discussion

Prediction of the 'no connection' state performed better than overall system prediction. This indicates that for other connectivity classes the system did not perform as well. For no connectivity there is only a single behavioral 'cause' in this scenario, namely jogging through the woods, where for others there were multiple causal patterns, e.g. leaving the house and emerging from the woods cause switching to UMTS. Here, we selected simple models for their ability to perform on the mobile device [4], and all behavioral patterns leading to the same connectivity change were modeled together. In reality, this generalization worsens prediction, and different types of behavioral patterns should be modeled separately.

In this work we have shown that observing behavior can allow a system to predict connectivity when temporal and location models fail. In general however, spacio-temporal models perform quite well, and can be effective in situations where behavior-based prediction would fail, e.g. predicting connectivity for jogging in the woods versus jogging through the city. Activity and behavior modeling is not a substitute for spatio-temporal modeling, and would probably perform worse in a generalized study. The work done here solely demonstrates that in situations where location and temporal models fail, behavior modeling can improve prediction. Integrating these different types of modeling and prediction is however not part of this work and is the subject of further research.

The system presented here uses explicit activity definitions and labels along with supervised machine learning to quantify human behavior (prediction training is unsupervised). In this work, supervised learning with explicit labels was used in order to quantify the performance of the reference implementation (f-score of 1 for labeled ground truth, v.s. 0.85 for recognized activities). The semantic and ontological meaning behind the activity labels is not required for, or used in, this process. This seems to indicate that the system could perform as well or better using unsupervised learning techniques, making it far more attractive for real world use as users must not provide any explicit input train the system. The system observes behavior and connectivity using unsupervised clustering techniques, and learns interdependencies.

7 Conclusion

Cloud computing has greatly enhanced the functionality and utility of pervasive mobile devices in recent years. Although initially the focus was on offloading data and processing to the cloud to support processes on the device, this process is now irreversible such that losing connectivity means many processes and applications cease to function. We have presented a concept which allows these processes to continue to perform offline, maintaining at least reduced functionality and utility. Our approach is to use prediction technology to inform processes and apps on the mobile device in advance of the disconnection events. We reason that behavior information is necessary in certain situations where location and temporal information do not suffice for predicting disconnection events. Our use case is based on a jogging scenario through a wooded area where the user has no connectivity for a period of time.

We first looked at how well our system performs if correct behavior information annotated explicitly by the subject is presented to the system. The results presented show that the hypothesis was correct, as offline periods were predicted 100% of the time with an average of about 8.5 minutes ahead of time, giving the system more than enough time to cache process state information from the cloud and continue operation locally. Furthermore, with a recall of 0.97, the amount of unnecessary caching was also reduced to a minimum.

We then evaluated how the system performs using behavioral information which it extracts from sensor signals using activity recognition techniques. Here we used an activity recognition toolchain which was able to correctly recognize behavior with an f-score of 0.85, containing non-negligible error. The results indicated that the system was still able to predict periods without connectivity 100% of the time, 8 minutes ahead of time on average, but with a drop in precision to 0.79, indicating an increase in unnecessary caching. We demonstrated that predicting disconnectivity in advance and caching necessary resources from the cloud can allow processes to be continued locally in an offline state. This preserves partial utility across the offline event horizon, reconciling the mobility of pervasive devices with the distributed nature of cloud computing.

Acknowledgment. This work was partially funded by the EIT ICT Labs.

References

1. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
2. de Araño, G.M., Pinto, A., Kaiser, J., Becker, L.B.: An evolutionary approach to improve connectivity prediction in mobile wireless sensor networks. Procedia Computer Science 10, 1100–1105 (2012), ANT 2012 and MobiWIS 2012
3. Forman, G., Zahorjan, J.: The challenges of mobile computing. Computer 27(4), 38–47 (April)

4. Gordon, D., Czerny, J., Miyaki, T., Beigl, M.: Energy-efficient activity recognition using prediction. In: 2012 16th International Symposium on Wearable Computers (ISWC), pp. 29–36 (June 2012)
5. Kobayashi, K., Matsunaga, Y.: Radio quality prediction based on user mobility and radio propagation analysis. In: International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 2137–2141. IEEE Computer Society Press, Los Alamitos (2009)
6. Kumar, K., Lu, Y.-H.: Cloud computing for mobile users: Can offloading computation save energy? Computer 43(4), 51–56 (2010)
7. Lungaro, P., Segall, Z., Zander, J.: Contextshift: A model for efficient delivery of content in mobile networks. In: 2010 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (April 2010)
8. Mayrhofer, R., Radi, H., Ferscha, A.: Recognizing and predicting context by learning from user behavior. Radiomatics: Journal of Communication Engineering, Special Issue on Advances in Mobile Multimedia 1(1), 30–42 (2004)
9. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Technical report (July 2009)
10. Motahari, S., Zang, H., Reuther, P.: The impact of temporal factors on mobility patterns. In: 2012 45th Hawaii International Conference on System Science (HICSS), pp. 5659–5668 (January 2012)
11. Mummert, L.B., Ebling, M.R., Satyanarayanan, M.: Exploiting weak connectivity for mobile file access. SIGOPS Oper. Syst. Rev. 29(5), 143–155 (1995)
12. Murphy, K.P.: Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). The MIT Press (August 2012)
13. Nicholson, A.J., Noble, B.D.: Breadcrumbs: forecasting mobile connectivity. In: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, MobiCom 2008, pp. 46–57. ACM, New York (2008)
14. Rahmati, A., Zhong, L.: Context-based network estimation for energy-efficient ubiquitous wireless connectivity. Mobile Computing 10, 54–66 (2011)
15. Seneviratne, A., Pedrasa, J., Rathnayake, U.: Network availability prediction: Can it be done? In: Global Information Infrastructure Symposium (2011)
16. Shin, C., Hong, J.-H., Dey, A.K.: Understanding and prediction of mobile application usage for smart phones. In: Proceedings of the Conference on Ubiquitous Computing, pp. 173–182. ACM, New York (2012)
17. Sigg, S., Gordon, D., von Zengen, G., Beigl, M., Haseloff, S., David, K.: Investigation of context prediction accuracy for different context abstraction levels. IEEE Transactions on Mobile Computing 11(6), 1047–1059 (June)
18. Ward, J.A., Lukowicz, P., Gellersen, H.W.: Performance metrics for activity recognition. ACM Trans. Intell. Syst. Technol. 2(1), 6:1–6:23 (2011)
19. Yang, Q., Zhang, H.H.: Web-log mining for predictive web caching. IEEE Trans. on Knowl. and Data Eng. 15(4), 1050–1053 (2003)