Network Prediction for Energy-Aware Transmission in Mobile Applications

Ramya Sri Kalyanaraman
Helsinki Institute for Information Technology HIIT
P.O. Box 15400, FI - 00076 Aalto
Espoo, Finland
ramya@hiit.fi

Yu Xiao, Antti Ylä-Jääski
Aalto University
P.O. Box 15400, FI-00076 Aalto
Espoo, Finland
{yu.xiao, antti.yla-jaaski}@tkk.fi

Abstract— **Network parameters such as signal-to-noise-ratio (SNR), throughput, and packet loss rate can be used for measuring the wireless network performance which highly depends on the wireless network conditions. Previous works on energy consumption have shown that the performance of wireless networks have impact on the energy efficiency of data transmission. Hence, it is potential to gain energy savings by adapting the data transmission to the changing network conditions. This adaptation requires accurate and energy-efficient prediction of the network performance parameters. In this paper, we focus on the prediction of SNR and the prediction-based network adaptations for energy savings. Based on the SNR data sets collected from diverse real-life networks, we first evaluate three prediction algorithms, namely, Autoregressive Integrated Moving Average, Newton Forward Interpolation, and Markov Chain. We compare these three algorithms in terms of prediction accuracy and energy overhead. Later we propose a threshold-based adaptive policy which controls the data transmission based on the predicted SNR values. To evaluate the effectiveness of using network prediction in adaptation, we use a FTP as a case study and compare the network goodput and energy consumption under different network conditions. The experimental results show that the usage of adaptations improves the network goodput. Furthermore, the adaptations using prediction can save up to 40% energy under specific network conditions when compared to the adaptation without prediction.**

Keywords-prediction; adaptation; SNR; power; context-awareness; policy-based.

I. INTRODUCTION

Mobile hand-held devices are more and more used for accessing rich multimedia content and various social media services. These new applications and services often call for more transmission capacity, processing power and high-quality displays which increase the energy usage. The improved user experience might be compromised by the short battery lifetime of the device. Unfortunately the improvements in battery technology are rather slow that there is a strong motivation to invest in software techniques to save the energy usage of mobile devices.

In network applications, major part of the energy is consumed by data transmission. The cost incurred during data transmission is not only dependent on the amount of data being transferred, but also the network goodput [2]. There is an indirect connection between the network goodput and other network performance parameters such as signal-to-noise-ratio (SNR). For example, in the wireless networks with higher SNR, the network goodput is usually higher, and therefore the energy consumption per a unit of transmitted data is lower. Hence, it is possible to save energy by transferring data only under the conditions where SNR values are high.

In this paper we focus on the prediction of SNR and its potential for improving the efficiency of an energy-aware network adaptation. With the help of the predicted future SNR values, it is possible to make a priori decisions as to when to schedule the data transmission over a wireless link. We propose a general solution of the prediction-based adaptive network transmission, including prediction algorithms of SNR and an adaptive policy. We evaluate our solution using FTP as a case study, while our solution is scalable to other network applications such as real-time streaming by taking the application-specific performance constraint into account in our adaptive policy.

Although the prediction of network parameters [3, 4] and their use in context-aware mobile applications have been discussed in the literature [5, 6, 7, 8, 9, 10], using the predicted network parameters for energy efficiency has not been widely studied, especially the usage of SNR for adaptive network transmission. Our contributions include the following aspects.

a) Comparing the performance of the prediction algorithms, namely, Auto Regressive Integrated Moving Average (ARIMA), Newton's Forward Interpolation (NFI) and Markov chain (MC). We analyze the performance in terms of prediction accuracy, processing load and energy overhead.
b) Proposing and evaluating a prediction-based power adaptation. The results show that the potential in energy saving under favorable network conditions can be up to 40% with the help of the predicted SNR.

This work is an extended version of our earlier publication [1]. Major portions of the experimental results have been updated, with enhancement in the offline training and online computation. In this work, we refine the evaluation by classifying the network conditions and analyzing the effectiveness of energy savings under different network conditions. In addition, the Linear Regression method used in [1] is replaced by MC, as MC shows better performance in short-term prediction.

The remainder of this paper is structured as follows. Section 2 reviews the related work in network prediction and

prediction-based adaptations. The prediction algorithms used in this study are introduced in Section 3, and the model fitting using these algorithms is described in Section 4. Section 5 describes a prediction-based adaptation. In addition, Section 5 includes the evaluation of the adaptation using FTP as a case study. In Section 6, we discuss the obtained experimental results. Finally, we present the concluding remarks and potential future work in Section 7.

II. RELATED WORK

Adaptive mobile applications are aware of and able to adapt to the changes in context such as the network signal strength, residual battery lifetime, and the user's location. The key for providing such adaptive applications lies in the decision-making of adaptations based on the available context data and predefined policies.

The context information includes the past, present, and future states of the contexts. Obviously, the past and present states of the contexts can be collected during runtime, whereas the future states are not available and have to be predicted based on the past and/or present states. The predicted states of the contexts are often used in context-aware applications for enhancing the effectiveness of adaptations, such as improving the resource utility and system performance. In [3] prediction of long range wireless signal strength is presented, where prediction is utilized for efficient use of resources in dynamic route selection in multi-hop networks.

For network-aware adaptive applications, network conditions are the most important contexts and can be described using signal strength, SNR, goodput, bit error rate and other contexts that reflect the status of the network transmissions. The so-called Box-Jenkins approach is the most widely used technique for network prediction. The basic idea is to train models with different parameters to fit the pre-collected data sets. This approach is adopted by many linear and non-linear models, such as ARIMA [13] and Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) [16]. In addition, Linear Regression [3], the MC [4], the Hidden Markov Model [7], and the Artificial Neural Network [17] have also been proposed for network prediction. In this paper, we apply ARIMA, MC and NFI [14] to the prediction of network SNR.

As mentioned before, network conditions can be described using various parameters. Different prediction algorithms might fit different contexts better depending on the characteristics of the contexts and the algorithms themselves. Previous researchers have been trying to apply different prediction algorithms to the prediction of different contexts. For example, [4] and [20] predict the status of network connectivity and the future location of mobile device using the MC, [3] utilized a linear regression approach for long range signal strength prediction, and [21] used the Hidden Markov Model to predict the number of wireless devices connected to an access point at a given time. Link prediction and path analysis using Markov chains in the World Wide Web are presented in [22].

Different prediction techniques are compared in terms of k-step-ahead predictability and performance overhead. For example, one-step-ahead predictability can be measured by Mean Square Error (MSE), Normal Mean Square Error (NMSE)[3], Root Mean Square Error (RMSE) [11] and Signal-to-Error Ratio (SER). The performance overhead includes the computational cost such as CPU cycles and memory access counts, and also the energy consumption caused by the prediction itself.

In most of the research work focusing on prediction for context-aware and adaptive applications, the motivation lies in increasing and analyzing the prediction accuracy of various prediction methods. In [11] various prediction methods such as Neural networks and Bayesian networks, state and Markov predictors are modeled for performing the next location prediction using the sequences of previously visited locations. In this paper, prediction accuracy has been considered as the main indicator for benchmarking the performance of various prediction techniques. To the best of our knowledge, our paper is the first one presenting the energy overhead of different prediction algorithms.

Adaptive applications purely based on policies consist of the condition and action pairs for different contexts. Here the possibility of anticipating the future trend of the context is not utilized. In adaptive applications with the support of prediction based adaptation, the system is given a chance to know the future trend of the context, and act accordingly.

Most of the existing adaptive network applications make adaptive decisions according to predefined policies. Predicted network contexts can be used as conditions of adaptive policies. In [18], the feasibility of implementing policy-based network adaptations in middleware architecture is presented. However, prediction-based network adaptations emphasizing possible energy conservation have not been widely studied. In this paper, we propose an adaptive file download in WLAN based on the prediction of network SNR, and compare the efficiency of prediction and adaptations among three different prediction techniques.

III. NETWORK PREDICTION ALGORITHMS

Dynamic adaptation to the environment and proactive behavior are key requirements for mobile adaptive applications. Prediction algorithms play a major role in providing a proactive nature for such applications [5]. In this paper we focus on predicting the network SNR that will be used for adaptive network applications.

In our approach, the nature of input data we observe to make the network prediction reflects the time series pattern [11]. We choose ARIMA [12], the most general model for forecasting a time series pattern. Selecting NFI [14] allows us to fit non-linear data over the curve. In addition, we experiment with MC representing a discrete time stochastic process. While evaluating the best suiting algorithm for our application, a trade-off is made between prediction accuracy, processing load and energy consumption, instead of compromising between the highest accuracy and the lowest energy consumption alone.

A. Auto Regressive Integrate Moving Average(ARIMA)

Classical linear time series models are based on the idea that the current value of the series can be explained as a function of the past values of the series and some other independent variables. ARIMA is one of the most famous linear predictive models used for network prediction. It is an integration of three models, namely, an autoregressive model of order *p*, a moving average model of order *q*, and a differencing model of order *d*. According to the definition in [18], a process, x_t, is said to be ARIMA(p, d, q) if

$$\varphi(B)(1-B)^d x_t = \theta(B) w_t \,, \tag{1}$$

where B is a backward shift operator, $\Phi(B)$ is an autoregressive operator, $\theta(B)$is a moving average operator, and w_t is assumed to be a Gaussian white noise. The three operators can be expressed as below.

$$B^d x_t = x_{t-d}, \tag{2}$$
$$\varphi(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p \big(\varphi_p \neq 0\big), \tag{3}$$
$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q \big(\theta_q \neq 0\big), \tag{4}$$

where $\varphi_1 \cdots \varphi_p$ and $\theta_1 \cdots \theta_q$ are constants.

Let $\hat{Y}(t)$ be the predicted SNR value at time *t*, and *Y(t-i)* be the observed SNR value at time $(t-i)$. The time series of SNR is considered to be an ARIMA (p, d, q) model if

$$\hat{Y}(t) = \mu + \sum_{i=1}^{d} Y(t-i) + \varphi \sum_{i=1}^{p} Y(t-i) - \theta \sum_{i=1}^{q} e(t-i) \,, \tag{5}$$

where $e(t-i)$ is the residual of prediction at time$(t-i)$, and μ is the intercept.

The values of p, d, q are identified based on the analysis of Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF), while the estimation of parameters θ, φ, μ are based on the least squares method which aims at minimizing the MSE of residuals as shown in Equation (6). The procedure of model fitting will be detailed in Section 4.

$$MSE = E(Y(t) - \hat{Y}(t))^2, \tag{6}$$

B. Newton's Forward Interpolation(NFI)

The observation at time *x* in a time series is defined as a function $f(x)$, and the values of *x* are tabulated at interval *h* as shown in Equation (4).

$$x = x_0 + i \times h, \tag{7}$$

where the first value of *x* is x_0, and *i* is the number of intervals between *x* and x_0. When only a few discrete values of $f(x), i = 0, 1, 2, \ldots$ are known, NFI aims at finding the general form of $f(x)$ based on known values by using the finite difference formulas.

Let $f_i \equiv f(x) \equiv f(x_0 + i * h)$, and the finite forward difference of a function f_i is defined by $\Delta f_i \equiv f_{i+1} - f_i$. Higher orders, such as the $j^{th} (j \leq i)$ order forward difference $\Delta^j f_i$, can be obtained by repeating the operations of the forward difference operator *j* times. For example,

$$\Delta f_0 = f_1 - f_0, \Delta^2 f_0 = \Delta(\Delta f_0) = f_2 - 2f_1 + f_0, \quad \Delta^3 f_0 = f_3 - 3f_2 + 3f_1 - f_0. \tag{8}$$

NFI model fits the observation at time *x* with an i^{th} degree polynomial as Equation (9).

$$f(x) = f_0 + (x - x_0)\frac{\Delta f_0}{h} + \frac{1}{2!}(x - x_0)(x - x_1)\frac{\Delta^2 f_0}{h^2} + \cdots + \frac{1}{(i-1)!}(x - x_0)(x - x_1) \ldots (x - x_{i-2})\frac{\Delta^{(i-1)} f_0}{h^{(i-1)}} + \frac{1}{i!}(x - x_0)(x - x_1) \ldots (x - x_{i-1})\frac{\Delta^i f_0}{h^i}. \tag{9}$$

When only the past *i* SNR observations are known, up to $(i-1)^{th}$ order forward difference can be developed. The prediction of SNR at time *x*, corresponding to the $(i+1)^{th}$ observation, is defined as a function *g(x)* in Equation(10).

$$g(x) = f_0 + (x - x_0)\frac{\Delta f_0}{h} + \frac{1}{2!}(x - x_0)(x - x_1)\frac{\Delta^2 f_0}{h^2} + \frac{1}{3!}(x - x_0)(x - x_1)(x - x_2)\frac{\Delta^3 f_0}{h^3} + \cdots + \frac{1}{(i-1)!}(x - x_0)(x - x_1) \ldots (x - x_{i-2})\frac{\Delta^{(i-1)} f_0}{h^{(i-1)}}. \tag{10}$$

Compared to *f(x)*, the error term *e(x)* is approximated as Equation (11).

$$e(x) \cong f(x) - g(x) \cong \frac{1}{N!}(x - x_0)(x - x_1) \ldots (x - x_{N-1})\frac{\Delta^N f_0}{h^N}. \tag{11}$$

Different from ARIMA, NFI could predict future values online directly based on past observations without offline training. Since the prediction accuracy of NFI depends on the size of N, the size of N can be selected by applying the least square method to the training data sets as used in ARIMA model fitting. The offline training and online implementation are described in Section 4.

C. Markov Chain(MC)

MC is a discrete time stochastic process with the Markov property, where given the present state, the future and past states are independent. To determine the future state from the present state, a probabilistic approach is used. A state space is defined where all the possible states of the system are represented. The change of state from one to the other is called state transition. Let us consider a discrete time process, X_n: n>0 with discrete state spaces {i, j}. The transition probability from i to j is computed as given in Equation (12).

$$P_{ij} = P(X_{n+1} = j \mid X_{n=i}). \tag{12}$$

The state transition involved in our model, and the online prediction of the state transition is explained in Section 4.

IV. NETWORK PREDICTION

Prediction of network related information adds value to ubiquitous applications, as the need for dynamic adaptation in a fluctuating network environment becomes a major requirement in such cases. In addition to network traffic and mobility prediction, predicting the value of SNR itself could provide a timely hint about possible changes in network conditions, and thus the application can prepare itself for adaptation.

In this section, we present the steps involved in offline training and online computation of prediction algorithms. The workflow is described in Figure 1. To perform offline training we carry out data collection. The spots chosen for data collection involve physical interferences and disturbances. Offline training enables us to estimate the parameters required for online computation, and the performance of prediction algorithms can also be analyzed using offline training. In online computation, the prediction algorithms are applied during runtime and the outcomes are matched with the adaptation policy.

A. Data Collection

The offline training used in our methodology significantly depends on the data collection. The data sets were gathered considering the following factors.

a) The user walked around a defined path, where the path involved different types of physical obstacles, interferences and noise.
b) A total of 12 data traces were collected at different times during three days, where the collection of each set lasted for 17 minutes.
c) The user walked around at a normal walking speed, and the signal strength and the noise level were recorded for every 100 millisecond.

We performed the data collection in the Computer Science department building of the university. Figure 2 shows the route map of the data collection. The defined path covered the first and second floors of the Computer Science department, which includes the corridor of the Data Communications Software lab, department library, café and the stairs to the second floor. To cover different network environments, we chose the path to include different types of physical obstacles, and interference. For example, the department corridor consists of physical obstacles such as wall structures and glass doors, and the library has metallic book shelves, and wooden structures. The open area in the cafe remains as a place with possible interferences due to multiple access points, and the significant presence of Bluetooth enabled devices.

B. Data Analysis and Pre-processing

By observing the collected data sets, we noticed that the SNR values remain unchanged for several milliseconds as shown in Figure 3. We therefore re-sampled the collected data sets at 1Hz, and used the new data sets for further processing. We chose 80% of the samples as the training data set, and the others for testing.

Data smoothing is often used for figuring out future trends, such as the trend in marketing size or stock prices. However, for short-term prediction like SNR prediction in our case, it remained an interesting question whether data smoothing still helped to improve the prediction accuracy. We therefore applied data smoothing over the training data set, and compared the prediction accuracy between the models using a smoothed data set and the original training data set.

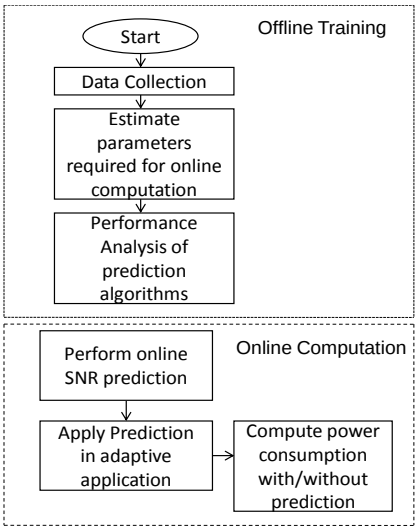

Figure 1: Workflow of offline training and online computation

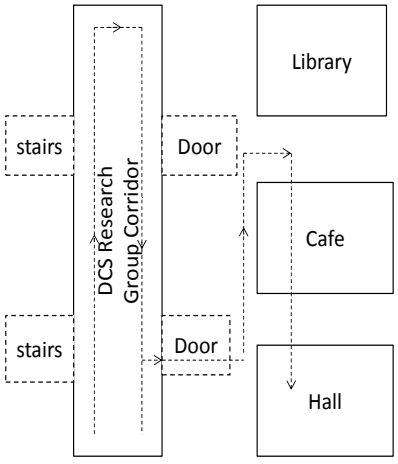

Figure 2: Route Map for Data Collection

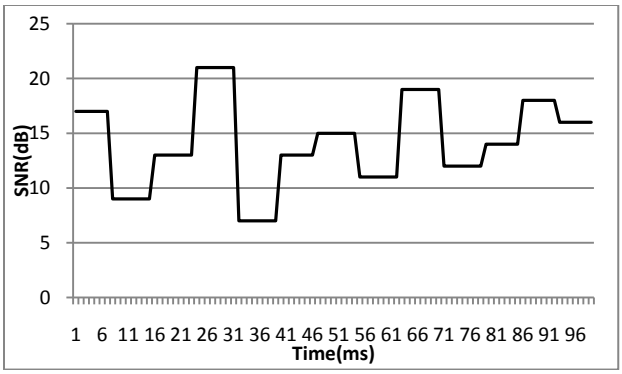

Figure 3: Samples of SNR values collected at 10Hz.

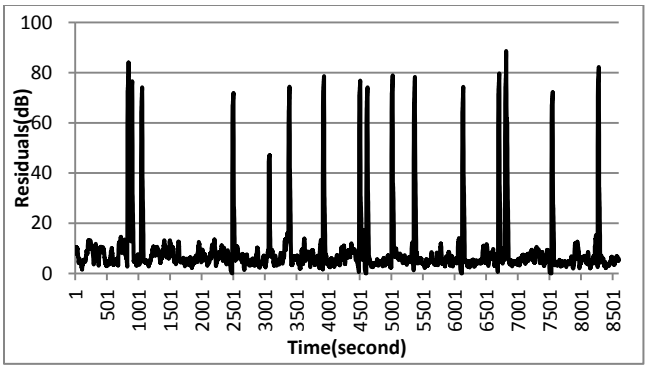

Figure 4: Residuals of smoothing with window size 20.

Table 1: Comparison of prediction accuracy among different models.

	MSE	NMSE	SER
ARIMA(0, 1,1)	97.58624	0.425374	7.456893
ARIMA(1, 0,1)	89.26294	0.389093	7.844067
ARIMA(2, 1,0)	174.4522	0.760430	4.934015
NFI(N=3)	692.8996	3.015758	-1.0487
NFI(N=4, 5)	99.3080	0.432879	7.380937

To perform data smoothing, we apply the simple Moving Average algorithm [23] which is used for time series data set smoothing in the field of statistics. According to the moving average algorithm, from the collected *N* data points, we perform averaging for every *W* data set, where *W* is often termed as the window size. The general expression for the moving average is given as in Equation (13).

$$M_t = \frac{[X_t + X_{t-1} + \cdots X_{t-w+1}]}{w}. \quad (13)$$

In general as the window size increases the trend of the smoothed data set becomes clearer, but with a risk of a shift in the function. To choose an optimal window size we performed smoothing with a varied window size, and selected the best one with the minimum MSE of the residuals. The residuals are the differences between the original values and smoothed values. According to our experimental results, smoothing with a window size of 20 has the minimum MSE which is 9.5 as shown in Figure 4.

C. Offline training/Model Fitting

1) ARIMA

ARIMA offline training is to fit an ARIMA(p, d, q) model to the collected data sets. The outputs of the model fitting includes the orders of autoregressive, differencing, and moving average, *p*, *d*, *q*, and the estimated parameter values of the autoregressive operator, the moving average operator, and the intercept, φ, θ, μ, as defined in Section 3.1. The model fitting includes the following five basic steps [18].

a) Data transform. We plotted the data and observed the possible data transform. As introduced in Section 4.1, we applied smoothing to the raw data, and trained the models using both data sets and comparing the results.

b) Model identification. The orders of autoregressive, differencing and moving average are identified by observing the ACF and PACF of the residuals. For orginal data sets, both ARIMA(0, 1, 1) and ARIMA(1, 0, 1) fit the data well and the differences in terms of the ACF and PACF of the residuals are negligible. Hence we trained both models and chose the better one after diagnostics. Similarly, for smoothed data sets with window size 20, ARIMA(2, 1, 0) fits better than other models for smoothed data sets.

*c) Parameter estimation.*We estimated the parameters of the expected models which obtain the minimum MSE.

*d) Diagnostics.*We applied the models obtained from the last step to the testing data sets and compared the accuracy among the models. First, we calculated the MSE, NMSE, and SER for each model. NMSE is a function of the MSE normalized by the variance of the actual data as defined in Equation (17).

$$NMSE = \frac{MSE}{E[(E[Y(t)] - Y(t))^2]}. \quad (17)$$

SER is defined as in Equation (18).

$$SER = 10log_{10}\left(\frac{E[Y(t)^2]}{E[(Y(t) - \hat{Y}(t))^2]}\right). \quad (18)$$

The smaller the MSE and NMSE are, the higher the accuracy is. Conversely, the bigger the SER is, the higher the accuracy is. The results of MSE, NMSE and SER for the three ARIMA models are listed in Table 1.

e) Model Selection. According to Table 1, ARIMA(2, 1, 0), with training based on the smoothed data set, showed the lowest accuracy compared to the models obtained from original data sets. ARIMA(1, 0, 1) fits the testing data sets better than others due to the smallest MSE and NMSE, as well as the highest SER. Hence we chose ARIMA(1, 0, 1) which is shown in Equation (14) for online SNR prediction.

$$\hat{Y}(t) = 22.1650 + 0.7997 * (Y(t-1) - 22.1650) - 0.0052 * e(t-1). \quad (14)$$

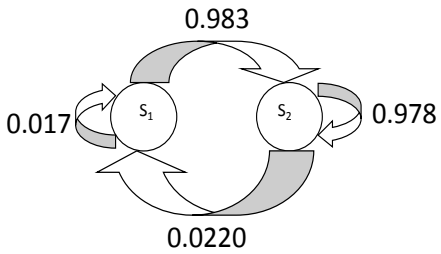

Figure 5: Transition probability between States, S_1 and S_2.

The implementation and online performance evaluation is presented in Section 4.3.

2) NFI

The accuracy of NFI prediction depends on the order of the forward difference *N* as explained in Section 3.3. We chose the size of N based on the least square method as used for ARIMA model fitting. As shown in Table 1, we compared the MSE when N is set to 3, 4, and 5 respectively. When N is 3, the MSE is much bigger than others, while the same predicted values and also the same MSE, NMSE, and SER values are obtained when either N= 4 or 5. Hence, we selected the order of the forward difference as 4 for online prediction.

3) Markov Chain

We designed a two-state Markov Chain model as shown in Figure 5, where S_1 and S_2 are the two possible states. Here S_1 represents the SNR value less than or equal to the threshold, and S_2 corresponds to a SNR greater than the threshold. Either S_1 or S_2 can represent the initial state depending upon the value of the SNR at the start of application.

There are four possible state transitions involved in the state machine, with four different transition probabilities. Let S_t and S_{t+1} represent the state of SNR at time t, and t+1. The four probable state transitions are as given below:

$$\begin{aligned}
SNR_{low,\,low}(t) &= Pr\{S_{t+1} = low \mid S_t = low\},\\
SNR_{low,\,high}(t) &= Pr\{S_{t+1} = low \mid S_t = high\},\\
SNR_{high,\,high}(t) &= Pr\{S_{t+1} = high \mid S_t = high\},\\
SNR_{high,\,low}(t) &= Pr\{S_{t+1} = high \mid S_t = low\}.
\end{aligned}$$

The values of the four probabilities are obtained from the training of the data sets. During the training, we set a threshold for discreetizing the samples into low and high states to 19, and calculated the probability distribution. The values of the four probabilities are shown in Figure 5. The precision of predicted outcomes using the Markov chain is 0.8191 as calculated according to Equation (19).

$$Precision = \frac{Number\ of\ correctly\ predicted\ values}{Total\ number\ of\ predicted\ values}. \qquad (19)$$

D. Online prediction

We present the online SNR prediction with the same monitoring and prediction frequencies in this Section, and set the initial values of the frequencies to 1Hz.

1) ARIMA(1, 0, 1)

The monitored and predicted SNR values are recorded in arrays s[] and p[] respectively. The pseudo code of ARIMA (1, 0, 1) online prediction is described as below.

```
//initialize the parameters of ARIMA(1,0,1);
Set ar=0.7997, ma=-0.0052, intercept=22.1650;
Initialize s[], p[] to 0;
i=0;
Repeat every 1 second{
Monitor SNR and write into s[i];

if (i>0){
        err = s[i-1]-p[i-1];
        p[i]=intercept + ar*(s[i-1]-intercept)
                +ma*err;
}
```

2) NFI

Set the order of forward difference to be 4. The monitored and predicted SNR values are recorded in array s[] and p[] respectively. The 1st, 2nd, and 3rd order of forward differences are calculated during runtime, and are written into arrays delta[], delta2[], and delta3[] respectively. The pseudo code of the online NFI prediction is described as below.

```
Initialize delta[], delta2[],delta3[],p[], s[] to 0;
i=0;
Repeat every 1 second{

Monitor SNR and write into s[i];

If (i>0)
        delta[i]=s[i]-s[i-1];
If (i>1)
        delta2[i]=delta[i]-delta[i-1];
If (i>2)
        delta3[i]=delta2[i]-delta2[i-1];

If (i>3)
    p[i]=s[i-4]+3*delta[i-3]+3*delta2[i-2]+delta3[i-
1];

        i++;
}
```

3) Markov Chain

The Markov chain SNR predictor computes the probable next state (S_{t+1}), given the current state (S_t) of SNR, according to the probability of different transitions obtained

Table 2: Comparison of performance and power overhead among different models.

	ARIMA	NFI	MC
CPU_CYCLES:100000	445	432	623
DCACHE_ACCESS_ALL:100000	31	25	73
Power(W)	0.365	0.366	0.474

Table 3. Classifications of the network conditions.

1	SNR mean is smaller than 15; SNR threshold is set to 15.
2	SNR mean is between 15 and 20; SNR threshold is set to 15 and 20.
3	SNR mean is bigger than 20, and standard deviation of SNR is smaller than 5. SNR threshold is set to 20.
4	SNR mean is bigger than 20, and standard deviation of SNR is not smaller than 5. SNR threshold is set to 15 and 20.

from offline training. The pseudo code for predicting the transition probability of the future state of SNR is given as follows:

```
//Monitor SNR and discreetize the received value.
Get current_SNR;

if (current_SNR <= Threshold)
        Current_state = S1;
else current_state = S2;

//Determine next state. SNRt[]is the transition matrix
obtained from offline training. SNRt[0] and SNRt[1]
represents the probability of the state transition from S1 to
S2 and vice versa .
Generate a random number, X.
if (current_state == S1)
{
        if (X ≤ SNRt[0])
                next_state = S1;
        else next_state = S2;
}
else
{
        if (X ≤ SNRt[1]
                next_state = S2;
        else next_state = S1;
}
```

D. Model Evaluation

We measured the overhead of online SNR prediction in terms of CPU cycle count, data cache access rate and power consumption. We ran oprofile [26] on the Nokia N810 while running the online prediction algorithms for 1 minute, and monitored the events, namely, CPU_CYCLES and DCACHE_ACCESS_ALL. The CPU cycle count and data cache access counts caused by the SNR prediction are listed in Table 2.

We used Nokia power measurement software to measure the power consumption during runtime. During measurements the display was turned off and the WLAN interface was turned on. The average power is 0.361 W when the system is idle and the power saving mode of WLAN is turned on, and 0.362 W when SNR monitoring is running at 1Hz at the same time. We measured the average power consumption when different online SNR predictors were running, and the comparisons are listed in Table 2.The energy overhead is calculated as the difference between the power consumption when the prediction algorithms are running and the power consumption of the device when it is in IDLE state. The power overhead caused by online prediction is between 0.004W and 0.113W depending on the prediction algorithm used.

V. PREDICTION-BASED NETWORK ADAPTATIONS

Wireless transmission is considered to cost much more energy than local processing on mobile devices [15], and the energy consumed by wireless transmission varies with the performance of the wireless networks. The performance is highly dependent on the network conditions. Generally, wireless transmission under better network conditions is more energy-efficient. Hence, adaptive mobile applications are expected to adapt wireless transmission that the transmission can be conducted under relatively good network conditions in order to save energy. In this section, we describe an application performing power adaptation using network prediction.

A. Overview

Based on the comparison and analysis of prediction algorithms made in Section 4 we are able to distinguish between the pros and cons of using different prediction methods, especially in terms of prediction accuracy, performance and energy overhead. After performing the offline training, and the analysis of different prediction methods we continue forward to the second phase of network prediction, online computation. In this section we present a real time user scenario that demonstrates the need for adaptation in ubiquitous applications. Realizing the requirements for network adaptation from the presented scenario we apply network prediction in the application.

Let us consider a practical scenario showcasing the need for prediction enabled adaptive applications.

Alice's mobile phone is connected to a WLAN network and she has been in the process of downloading a file. The network quality associated with the device fluctuates between poor to strong. The application with the help of prediction foresees the status of network, and hence chooses an adaptive action accordingly in order to gain the optimal power saving. The adaptive action can be pausing, stopping, or continuing the file download.

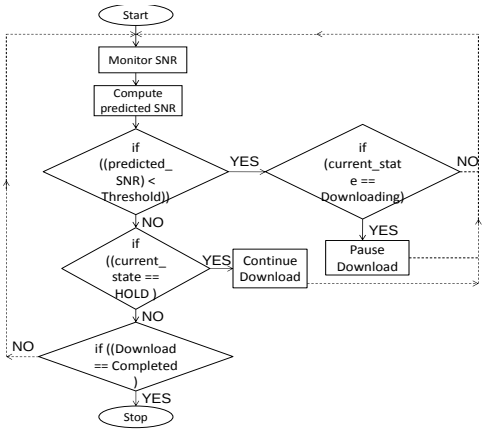

Figure 6: Flow chart representing online adaptation

B. Adaptation Policy

Adaptation policy defines the actions to be performed based on the changing context data. Adaptation is triggered in mobile applications as the outcome of the prediction is matched with a simple predefined adaptation policy. The adaptation policy can be defined by either the application developer or end-user. In the case of the end-user, policies can be in the form of user preferences for adaptive applications. The priority and conflict management between prediction results and current network condition however are outside the scope of this paper.

In the file download scenario, we designed an adaptive policy which enables the application to pause or continue the file download depending on the predicted future SNR. The basic idea is to transfer the data only when the SNR is above a threshold. We classify the network condition into four types using the mean and standard deviation of SNR as described in Table 3. The threshold value is set depending on the type of the network condition. It is described as below.

if (predicted_SNR < threshold) && (current_state == downloading)
Pause download;
else if ((predicted_SNR ⩾ threshold) && (current_state == hold))
Continue download;
else
// Continue the same action state.
Continue download | Pause download;

While applying the above mentioned adaptive policy for the applications such as real-time streaming which is not delay-tolerant, the settings of the threshold value must take the tolerant delay into account. For example, the delay caused by the pause operation can be predicted based on the predicted SNR value. Only when the delay is tolerable, the download can be paused. The prediction based adaptation continues till the file download is completed. Figure 6 describes the online adaptation performed in the file download scenario using the online prediction and defined policy. The implementation for online computation of the prediction forecast is combined with the adaptation code. Thus the prediction outcome is directly used for choosing the adaptation policy, and executes an adaptive operation in the file download application.

C. Experimental results

The prediction-based power adaptation is evaluated in terms of energy consumption, network goodput, and prediction accuracy, under different network conditions. For different network conditions, we test the adaptations with an SNR threshold of 15 and/or 20 with an exception for the MC method. For the MC method we set the threshold to be 19, as the offline training and online computation has to follow the same threshold value. Hence, our definitions of test cases include the network conditions in terms of the mean and standard deviation of SNR values, SNR threshold value, and possible network goodput range. Though the MC method follows a different threshold, the measurements obtained using the MC as the prediction method can still be accommodated within the test cases described. Based on the analysis of our experimental network conditions, we divide our test cases into four scenarios as listed in Table 3.

The adaptation code is implemented using C language. The power measurements for the adaptive file download with and without prediction were carried out on a public 802.11g network in our campus with the power saving mode on. A TCP client *wget* (http://www.gnu.org/software/wget/) running on the N810 downloaded a MySQL software package from a remote file server [26]. The file name is MySQL-shared-compat-5.1.42-0.rhel3.x86_64.rpm with a size of 5072982 Bytes. We set both the prediction and monitoring frequency to 1Hz and performed one-step-ahead prediction. Adaptation is performed for each prediction output, which means the adaptation policy is executed every one second.

We measured the power consumption in different scenarios without adaptation, and used the values as base line for comparison. The power measurement results are listed in Table 4. Since the energy consumption depends on the network goodput, and the network goodput can vary even in the same scenario, we list multiple samples in each scenario with possible network goodput. The testing results of each scenario are listed in Table 5.

The download duration is the duration from the beginning to the ending of the file download, while the total energy includes the energy consumption of the mobile device during the download duration. Pause duration is the total duration when the file download is paused due to power adaptation, and the actual download duration is the download duration deducted by the pause duration. We calculate the network goodput as the downloaded file size divided by the actual download duration. The prediction accuracy is the ratio of the number of correctly predicted samples to the total number of predicted samples. For example, if the predicted SNR is higher than the SNR threshold and the real SNR is lower than the SNR threshold, or if the predicted SNR is lower than the SNR threshold while the real one is higher, it

Table 4: Baseline of energy consumption in different network scenarios.

Type	SNR mean	SNR standard deviation	Energy(J)	Network Goodput (KB/s)
1	14.245	5.806	92.075	54.590
1	11.391	5.454	118.945	36.765
2	19.9	3.872	52.903	158.521
2	19.346	6.425	43.943	178.526
2	15.660	5.010	45.918	165.136
3	24.273	4.548	38.990	225.196
3	26.690	4.635	55.213	147.878
3	26.952	3.290	63.745	120.828
4	22.325	9.024	62.775	60.047
4	21.900	8.110	45.813	176.925
4	24.204	5.500	194.58	18.264

Table 5: Experimental results for test cases under different network conditions.

Type	SNR mean	SNR standard Deviation	Download duration(s)	Pause duration(s)	Network goodput (KB/s)	Energy(J)	Accuracy (%)	Model	Threshold
1	13.450	4.625	78.751	44	142.559	68.463	85.00	ARIMA	15
1	13.310	6.468	101.750	37	76.511	55.200	82.76	ARIMA	15
1	14.926	7.387	54.001	25	170.825	59.296	77.78	NFI	15
1	12.615	6.446	64.748	38	185.213	60.523	87.02	NFI	15
2	17.567	6.026	40.499	12	173.834	49.098	78.38	ARIMA	15
2	16.514	8.862	35.249	15	244.658	45.47	83.78	NFI	15
2	17.283	4.662	133.999	90	115.298	108.733	92.54	ARIMA	20
2	18.113	5.106	81.999	53	170.836	59.590	86.25	NFI	20
2	18.901	4.867	60.000	31	170.800	59.718	78.43	MC	19
3	27.864	4.560	22.001	1	235.898	39.030	90.91	ARIMA	20
3	25.519	2.513	22.000	0	225.186	39.970	100	NFI	20
3	24.296	3.831	30.501	4	190.538	46.87	74.07	MC	19
4	34.206	7.014	32.251	0	153.61	38.213	100	ARIMA	15
4	30.842	8.772	37.999	3	141.549	44.068	89.47	NFI	15
4	28.926	11.198	30.251	9	233.122	38.423	96.43	ARIMA	20
4	26.758	6.083	30.748	3	178.538	47.648	100	NFI	20

Table 6: Experimental results with low network goodput.

SNR mean	SNR standard deviation	Download duration(s)	Pause duration(s)	Network goodput(KB/s)	Energy(J)	Accuracy (%)	Model	Threshold
28.436	5.543	276.250	1	17.998	189.645	98.91	ARIMA	15
21.089	8.971	216.509	90	40.1	137.753	87.68	ARIMA	20
24.5	7.104	195	50	34.165	137.617	70.97	MC	19

is considered to be a wrongly predicted sample.

In the network scenarios of Type 1, SNR fluctuates between 0 and 20 with the mean less than 15. When the threshold is set to 15, the file download is paused when the predicted values are lower than 15. According to Table 5, the pause durations take around 36% to 59% of the download duration, whereas the network goodput increase more than 100% compared to the base line. The energy consumption with adaptation is reduced by 40% on average.

For Type 2, when the threshold is set to 20, the pause duration becomes very big. The overhead caused by the adaptation includes the energy overhead of prediction itself and the energy cost during the pause duration. The total overhead is bigger than the energy savings caused by the adaptations. Hence, when the SNR is relatively stable with the standard deviation less than 5 and the mean value bigger than 15, it is not energy-efficient to adopt adaptation with a threshold of 20 even though the prediction accuracy is relatively high. When the threshold is set to 15, the download duration decreases and thus leads to energy savings when compared with the corresponding Type 2 scenario.

For Type 3, the SNR values are high and the standard deviation is very small. The pause duration is very short, and the impact on network goodput is negligible. The energy consumption depends on the network goodput, and the energy overhead is caused by the prediction algorithm, which is around 0.1W. For type 4, when the threshold is set to 15, it leads to energy savings, whereas when the threshold is set to 20 the energy saving is comparatively less.

Due to the workload of the file server and network overload, the network goodput sometimes becomes much smaller under the same SNR conditions. As shown in Table 6, for Type 4, when network goodput is relatively lower, the energy consumption is more than 190J. Compared to the similar case in Table 4, the adaptation can help save energy especially when the threshold is set to 20.

VI. DISCUSSION

It is well-known that network transmission at a higher data rate results in less energy consumption. Our adaptation aims at adapting the network transmission to network conditions in terms of SNR values by controlling the transfer operations depending on the one-step-ahead SNR prediction. In our adaptation based on the prediction method, if the predicted SNR value is lower than a predefined threshold, transmission will be paused or else continued. Through such adaptation, network goodput could be increased or maintained since the download speed is relatively higher under better network conditions in terms of higher SNR values. In addition, it is possible to reduce some unnecessary retransmission causing high loss rate in a noisy network environment.

The effectiveness of our adaptation depends on the trade-off between the energy overhead caused by the SNR prediction and the energy savings made by increased network goodput. The former one is considered to be stable and independent of network conditions, whereas the latter one varies with network scenarios. To figure out the impact from different network scenarios on the effectiveness of our adaptation, we divide the experimental network scenarios into 4 types based on the mean and standard deviation of SNR values. The type classification is mainly due to the fact that in real time network measurements it is hard to get a stable SNR value.

According to our evaluation results presented in Section 5, when SNR values are generally low as in Type 1, the increase in network goodput is significant, and hence the adaptation is profitable. In the scenarios where the SNR values fluctuate heavily even though the mean of SNR is high, such as Type 4, our adaptation also could save energy to a certain extent. If the network conditions are relatively stable, for example, in Type 2 and 3, when the standard deviation is less than 5, there is not much advantage for the threshold-based adaptation. Energy consumption could not be saved, but is wasted due to the energy overhead caused by network adaptations. In addition to the SNR range, the selection of threshold has an impact on the effectiveness of our adaptation. For example, in the network conditions of Type 2, when the threshold is set to 20, the pause duration is close to 0. However, when the threshold is changed to 15, the adaptation becomes more energy efficient. Hence, in summary, threshold-based adaptation is energy-efficient when network conditions fluctuate in a big range or remain in a relatively bad state. In other words, when SNR values are generally low, such as lower than 15, or when the standard deviation of SNR values is big, network adaptations help save energy up to 40%.

In this paper, we compare the prediction accuracy of different prediction algorithms during offline and online experiments. Our experimental results show that all the three algorithms could attain reasonable high prediction accuracy, and that the energy savings depend more on network conditions than prediction accuracy in our case. In the case that the predicted values are smaller or higher than the real values, it does not always follow that there is an impact on the results of adaptations. For example, when the wrong prediction causes an unnecessary pause of the download, or does not pause the download when the SNR value is very low, it might increase the download duration and waste energy. When both predicted and real values are bigger or smaller than the threshold, it does not change the adaptive operation.

Among the three prediction algorithms, ARIMA has relatively higher prediction accuracy, and NFI performs better than MC. However, ARIMA requires offline learning. Hence in a new network environment which has not been trained, the accuracy might be lower and the energy savings might be reduced too. To choose the prediction algorithm to use for prediction, it would be better to apply ARIMA if the user's moving paths could be predefined. For the application scenarios where the network conditions are totally new to mobile devices, it is wiser to choose NFI which can adjust itself to the new network scenarios quickly.

VII. CONCLUSION AND FUTURE WORK

To summarize our contribution in this paper, we have explored the possibility of applying network predictions to network-based power adaptation, and implemented three prediction algorithms, ARIMA, NFI, and the MC in adaptive file transfer on mobile devices. We compared the effectiveness of the algorithms while taking the performance of prediction like prediction accuracy and performance overhead into account. The experiments were conducted in a number of settings in public WLANs.

From the lessons learned through prototype level implementation and experimental evaluation, we have figured out the future roadmap for our work that helps to improve the prediction and adaptation techniques used. The temporal difference (TD) method which is often used for reinforcement learning can be a potential topic to be investigated for enhancing our prediction techniques. The advantage of utilizing the TD method is the higher prediction accuracy, and it requires less memory for computation [24]. In addition, TD is applicable for multi-step prediction [25]. As indicated in Section VI the threshold based adaptation could lead to more energy savings, and we will explore the possibility of using dynamic threshold values. The dynamic threshold especially could be helpful in situations where the network behavior fluctuates.

ACKNOWLEDGMENTS

This work was supported by TEKES as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology, and Innovation in the Field of ICT). We also extend our thanks to Chengyu Liu for

implementing the prediction algorithms, and Gopalacharyulu, PV for providing valuable suggestions while choosing prediction algorithms.

REFERENCES

[1] R. Sri Kalyanamaran, Y. Xiao, and A. Ylä-Jääski, "Network Prediction for Adaptive Mobile Applications", UBICOMM'09: In Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies, pp. 141 – 146, Malta, October 2009.

[2] Y. Xiao, P. Savolainen, A. Karpanen, M. Siekkinen, A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications", e-Energy'10: In Proceedings of the 1st International Conference on Energy-efficient Computing and Networking, pp. 75 – 84, Passau, Germany, April, 2010.

[3] X. Long and B. Sikdar, "A Real-time Algorithm for Long Range Signal Strength Prediction in Wireless Networks", WCNC 2008: In proceedings of IEEE Wireless Communications and Networking Conference, pp. 1120 – 1125, 2008.

[4] Y. Vanrompay, P. Rigole, and Y. Berbers, "Predicting network connectivity for context-aware pervasive systems with localized network availability", WoSSIoT'07: In Proceedings of 1st International Workshop on System Support for the Internet of Things, Lisbon, Portugal, March, 2007.

[5] R. Mayrhofer, "An Architecture for Context Prediction", In Advances in Pervasive Computing, Volume 176, Austrian Computer Society (OCG), pp. 65 – 72, 2004.

[6] C.Anagnostopoulos, P. Mpougiouris, and S. Hadjiefthymiades, "Prediction intelligence in context-aware applications", In Proceedings of the 6th international conference on Mobile data management, MDM'05, pp. 137 – 141, 2005.

[7] Y. Wen, R. Wolski, and C. Krintz, "Online Prediction of Battery Lifetime for Embedded and Mobile Devices", In Proceedings of 3rd International Workshop on Power-Aware Computer Systems, pp. 57 – 72, December, 2003.

[8] A. Gupta and P. Mohapatra, "Power Consumption and Conservation in WiFi Based Phones: A Measurement-Based Study", SECON'07: In Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks, pp. 122 – 131, June, 2007.

[9] M . Vukovic, I. Lovrek, and D. Jevtic, "Predicting user movement for advanced location-aware services", SoftCom: In Proceedings of 15th International Conference on Software, Telecommunications, and Computer Networks, pp. 1 – 5, 2007.

[10] D. Narayanan, J. Flinn, and M. Satyanarayanan, "Using History to Improve Mobile Application Adaptation", WMCSA '00: In Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications, pp. 31 – 40, 2000.

[11] J. Petzold, F. Bagci, W. Trumler, and T. Ungerer, "Next Location Prediction Within a Smart Office Building", ECHISE'05: In proceedings of 1st International Workshop on Exploiting Context Histories in Smart Environments at the 3rd International Conference on Pervasive Computing, Munich, Germany, May, 2005.

[12] A. S. Weigend, and N. A. GershenField, "Time Series Prediction: Forecasting the Future and Understanding the Past", SFI Studies in the Sciences of Complexity, Proc. Vol XV, Adisson-Wesley, 1993.

[13] G. E. P Box, and G. M. Jenkins, "Time Series Analysis: Forecasting and Control". San Francisco: Holden Day, 1970, 1976.

[14] F. B. Hildebrand, "Introduction to Numerical Analysis" (2nd edition) McGraw-Hill. *ISBN 0-070-28761-9*, Pp 43-59, 1974.

[15] K. C. Barr and K. Asanovic, "Energy-aware Lossless Data Compression", ACM Transactions On Computer Systems. Vol. 24, No. 3, pp. 250 – 291, August, 2006.

[16] Y. Hao, L. Chuang, S. Berton, and L. Bo, and M. Geyong, "Network traffic prediction based on a new time series model", International Journal of Communication Systems, Volume 18, Issue 8, pp. 711–729, October, 2005.

[17] N. Sadek, and A. Khotanzad, "Multi-scale network traffic prediction using k-factor Gegenbauer ARMA and MLP models", AICCSA'05: In Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications, 2005.

[18] J. Z. S. Tenhuen, and J. Sauvola, "CME: a middleware architecture for network-aware adaptive applications", PIMRC'03: In Proceedings of the 14th International Symposium on Personal, Indoor, and Mobile Radio Communications, , pp. 839 – 849, Beijing, China, Septemer, 2003.

[19] R. H. Shumway, and D.S. Stoffer, "Time Series Analysis and Its Applications: With R Examples", Springer Texts in Statistics, 2nd Edition, pp. 85 – 154, 2006.

[20] D. Katsaros, and Y. Manolopoulos, "Prediction in wireless networks by markov chains", IEEE Journal on Wireless Communications, Volume 16, No. 2, pp. 56 – 63, 2009.

[21] MD. O. Gani, H. Sarwar, and C.M. Rahman, "Prediction of state of wireless network using markov and hidden markov model", Journal of Networks, Volume 4, No. 10, pp. 976 – 984, 2009.

[22] R. R. Sarukkai, "Link prediction and path analysis using markov chains", In Proceedings of the 9th International World Wide Web Conference on Computer Networks, pp. 337 – 386, 2000.

[23] Y. L. Chou, "Statistical Analysis", Holt International, 1975

[24] R. S. Sutton, "Learning to predict by the Methods of Temporal Differences", Journal for Machine Learning, Volume 3, Number 1, pp. 9 – 44, Springer Netherlands, August, 1988.

[25] X. Z. Gao, S.J. Ovaska, and A.V. Vasilakos, "Temporal difference method-based multi-step ahead prediction of long term deep fading in mobile networks", In Computer Communications, Volume 25, Issue 16, pp. 1477 – 1486, October, 2002.

[26] OProfile, System profiler for Linux, http://maemo.org/development/tools/doc/chinook/oprofile/

[27] MYSQL Software Download Site, http://vesta.informatik.rwth-aachen.de/mysql/Downloads/MySQL-5.1