Cross-layer reduction of wireless network card idle time to optimize energy consumption of pull thin client protocols

Pieter Simoens, Farhan Azmat Ali, Bert Vankeirsbilck, Lien Deboosere, Filip De Turck, Bart Dhoedt, Piet Demeester, Rodolfo Torrea-Duran, L. Van der Perre, A. Dejonghe

***Abstract*—Thin client computing trades local processing for network bandwidth consumption by offloading application logic to remote servers. User input and display updates are exchanged between client and server through a thin client protocol. On wireless devices, the thin client protocol traffic can lead to a significantly higher power consumption of the radio interface. In this article, a cross-layer framework is presented that transitions the wireless network interface card (WNIC) to the energy-conserving sleep mode when no traffic from the server is expected. The approach is validated for different wireless channel conditions, such as path loss and available bandwidth, as well as for different network roundtrip time values. Using this cross-layer algorithm for sample scenario with a remote text editor, and through experiments based on actual user traces, a reduction of the WNIC energy consumption of up to 36.82 % is obtained, without degrading the application's reactivity.**

***Index Terms*—wireless thin client, energy optimization, cross-layer, power consumption**

I. Introduction

Thin client computing refers to the paradigm where the client relies on a remote server to perform a significant fraction of its computational tasks. User applications are executed on a remote server and the thin client device only deals with user interaction and rendering of the screen graphics. The principle is illustrated in Figure 1.

Especially in a mobile context, the thin client concept is very promising. Mobile devices often lack the required processing resources to execute demanding applications locally and users need to resort to stripped-down versions tailored to the specifics of the mobile phone operating system. In the thin client computing model, the applications are executed on server farms typically equipped with important computation power, whereas at the client only a small-footprint viewer is required to render the display updates received from the server. Through thin client computing, even demanding applications, requiring specific hardware such as graphical processing units, can be accessed from resource-constrained mobile devices. Furthermore, since only basic client functionality and processing power are required, thin client devices can be made lightweight and potentially energy efficient, two characteristics that are highly appealing to mobile users.

P. Simoens, F. Azmat Ali, B. Vankeirsbilck, L. Deboosere, F. De Turck, B. Dhoedt and P. Demeester are with the Dept. of Information Tech., Ghent University, Belgium. R. Torrea-Duran, L. Van der Perre and A. Dejonghe are with the Interuniversity Micro Electronics Center, Leuven, Belgium. e-mail: pieter.simoens@intec.ugent.be.

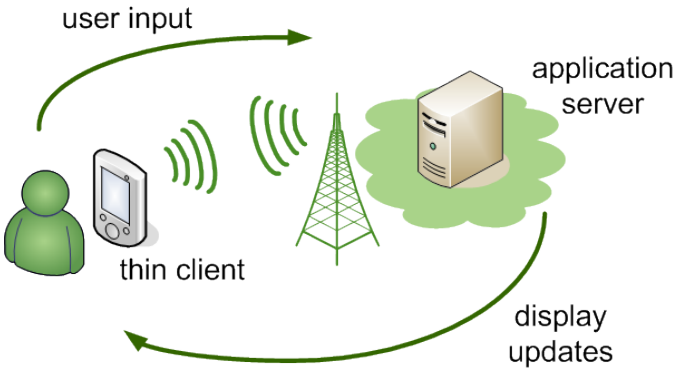


Fig. 1. In thin client computing, the client offloads a major part of its processing tasks to a remote server. User input and display updates are communicated over a thin client protocol. On mobile devices, the data exchange might result in significant WNIC energy drains from the battery.

Compared to the standard client-server computing model, thin client systems place a large traffic load on wireless links [1]. When the conventional approach is followed, i.e. applications are run locally, network communication is typically limited to storage or retrieval requests. On the other hand, a thin client must be connected to the network at all times to transmit user input and receive display updates. This data exchange might result in a significant energy drain from the device battery by the wireless network interface card (WNIC). Although the exact share in the total power budget depends on the specific hardware and the volume of data exchanged, it is generally acknowledged that the WNIC is a significant source of consumed energy on mobile devices [2]. For example, when streaming video to a Compaq iPaq, the WNIC amounts for 37.7 % of the total power consumption [3]. In typical smartphone usage, the Wi-Fi interface accounts for 25 % of the total power budget [4]. On an Asus EEE PC, a device without hard disk, merely switching on the WiFi interface without actually transmitting or receiving data, already increases the total device power consumption by 12 % [5]. On thin clients, reducing energy consumption is even more important because the available battery capacity is inherently limited by the small form-factor constraints of such devices. Clearly, there is a need for energy optimization schemes to mitigate WNIC energy consumption on thin clients.

The WNIC can be in four possible states, dependent on

the WNIC components that are active: send, receive, idle or sleep mode. On thin clients, a significant fraction of the WNIC energy consumption is due to the time spent in idle mode [6]. Even in scenarios with important downstream traffic due to complex display updates, still half of the total WNIC energy consumption is due to the time spent in idle mode. As the time spent in send and receive mode is limited, only minor energy gains are to be expected from existing algorithms that optimize encoding and transmission parameters. On the other hand, potentially major energy reductions can be expected by putting the WNIC in the energy-conserving sleep mode during idle intervals. Indeed, dependent on the hardware tested, a WNIC consumes 5 to 10 times less energy in sleep mode as compared to the idle mode [7], [8], [9].

In idle mode, the terminal is only detecting incoming frames so it is ready to shift quickly to another mode. This mode is used during periods when the terminal keeps synchronization and receives broadcast information. In sleep mode, most WNIC components are turned off, including the radio interface, and the device cannot determine whether data is being transmitted to it. The WNIC sleep intervals must be carefully chosen in order not to miss any data that is destined for this device. However, information on the arrival time of expected downstream traffic is only available at the highest layer of the protocol stack. For example, only the highest layers can distinguish transmitted data requests from TCP acknowledgements. On the other hand, data transmission on the wireless channel is controlled by the medium access control layer (MAC), and only this layer can instruct the WNIC to go into sleep mode. Hence, in order to reduce the energy consumption by converting idle time to sleep time without risking to lose any downstream data, a cross-layer approach is required. WNIC sleep opportunities are identified at the thin client protocol layer and communicated to the MAC layer.

In this paper, we propose a cross-layer algorithm that runs at the client and that is completely transparent to the server. The thin client protocol messages exchanged between client and server are analyzed to forecast the arrival time of display updates transmitted by the server. The presented algorithm is applicable to client-pull thin client protocols and for all applications that only update their display after some user interaction, such as text editing or web browsing. When no traffic from the application server is expected, the WNIC of the thin client is put in sleep mode to conserve energy. Throughout this paper, we adopt the scenario of a user that is connected to the application server over a 802.11 Wi-Fi link. Despite the increasing popularity of 802.11 wireless interfaces in home networks, public places and company buildings, the 802.11 energy consumption remains an important challenge [10]. The efficiency of the cross-layer algorithm has been validated for a wide range of wireless channel conditions. Furthermore, simulations have been conducted based on actual user input traces, clearly demonstrating how the cross-layer framework successfully reduces the client energy consumption while preserving the client perceived latency.

The remainder of this paper is structured as follows. In section II, related work is outlined that contributes to optimizing energy efficiency for thin clients. Section III details the specific client-server communication patterns of a thin client protocol, demonstrating the origin of the idle intervals. The cross-layer framework itself, as well as the idle time reduction algorithm, are detailed in section IV. In section V, experimental results are presented to validate the approach.

II. Related work

To tackle energy consumption of wireless devices, many cross-layer approaches have been published that involve all layers of the protocol stack. An extensive overview of cross-layer approaches at the MAC/PHY layers is given in [11]. At the network layer, research was mainly devoted to cross-layer routing algorithms for wireless mesh networks [12], [13]. Exploiting TCP protocol layer information for energy efficiency is presented in [9]. By deploying a proxy agent at the base station, the time required for retransmitting packets lost at the wireless link is reduced, resulting in potential longer sleep times. At the application layer, previously presented cross-layer techniques are focused on an optimization of encoding and transmission strategies. In [14], video codec parameters are configured according to the current status of the wireless channel. In [15], voice packets that can tolerate moderate packet losses are tagged by the application layer. MAC layer acknowledgements are disabled for these packets, in order to reduce the energy consumed to transmit and receive the voice data.

All these cross-layer energy saving approaches are mainly targetting an improved transmission scheme to reduce the time spent by the WNIC in send and receive mode. However, as pointed out in section I, only limited energy gains can be expected for thin clients from such data transmission optimization mechanisms. Energy optimization schemes for thin clients should be primarily focused on converting WNIC idle time to sleep time. In this respect, relevant work on optimizing WNIC power consumption for web-based applications is presented in [10]. The authors observe how users downloading data from remote sites generate a bursty data traffic pattern. Consecutive bursts are interleaved by user think times, during which the WNIC is put in sleep mode. However, the authors assume that each data burst is triggered by a client request and that user think times are in the order of minutes. Both assumptions do not hold for the thin client case. First, in thin client protocols not only requests are sent, but an important fraction of the data sent upstream are user events. As will be explained in more detail in section III, not every user input immediately results in a new display update being generated by the server. Second, the timescale of subsequent user input is most often in the order of milliseconds, rather than in the order of minutes. Several key strokes or pointer position updates can be generated within one second.

In [16], user input is monitored and exploited to achieve an interaction-aware energy management scheme for WNICs. The proposed mechanism correlates user interactions with resulting network activity to predict future levels of I/O demand. More specifically, during a learning phase of the prediction algorithm, mouse events are captured at the application layer and

combined with the monitored network traffic. This information is used to transition the WNIC to a higher energy state before data arrives. Our approach enchances this mechanism, as it requires no specific training but is continuously monitoring the exchanged protocol traffic.

In [2], the authors shape a TCP stream into bursts to increase potential sleep intervals by manipulating the TCP congestion window. Furthermore, the client transmits additional probe packets to detect the end of a burst. The approach is complimentary to the cross-layer framework presented in this article, as it is based on TCP protocol layer information, whereas our framework exploits thin client protocol layer information. However, our approach achieves higher energy gains, because the thin client protocol layer contains more semantic information, e.g. it is aware when the server might send a new display update. Furthermore, no additional probe packets are required.

III. Thin client protocol operation

The energy saving approach that is presented in this article, is based on an identification of intervals during which no display updates from the server are expected. Therefore, in this section, details are given on how the server schedules the transmission of display updates.

A. Thin client protocol modes

Thin client protocols can operate either in server-push or client-pull mode [17]. In the server-push model, the server determines autonomously when to send a display update to the client. In the client-pull model, the client sends an explicit request to receive a new display update. Whereas protocols operating in the server-push model might overwhelm the client with display updates, protocols following the client-pull model might exhibit higher upstream bandwidth requirements due to the requests for display updates.

The design of an energy saving approach for server-push protocols, such as Microsoft Remote Desktop Protocol (RDP) or Citrix Independent Computing Architecture (ICA) requires less algorithm logic compared with client-pull protocols. In server-push architectures, the server could be configured to transmit display updates at a fixed rate. Consequently, to save energy, the client only needs to put the WNIC in sleep mode for a fixed period each time an update is received. In case the server adjusts the push rate, e.g. to the current network status, it would still suffice to inform the client on the new update frequency to allow it to resynchronize the sleeping periods. We have previously developed a framework that dynamically adjusts the update frequency of push thin client protocols and synchronizes this value with the client [18]. In this article, we focus on energy saving with pull thin client protocols. Compared with server-push thin client protocols, the identification of idle intervals is more complicated because the period between subsequent display update is more variable. To validate the approach, we will use the Virtual Network Computing architecture as open source implementation of client-pull protocols. The proposed algorithm can however be directly applied to other client-pull protocols.

B. Virtual Network Computing architecture

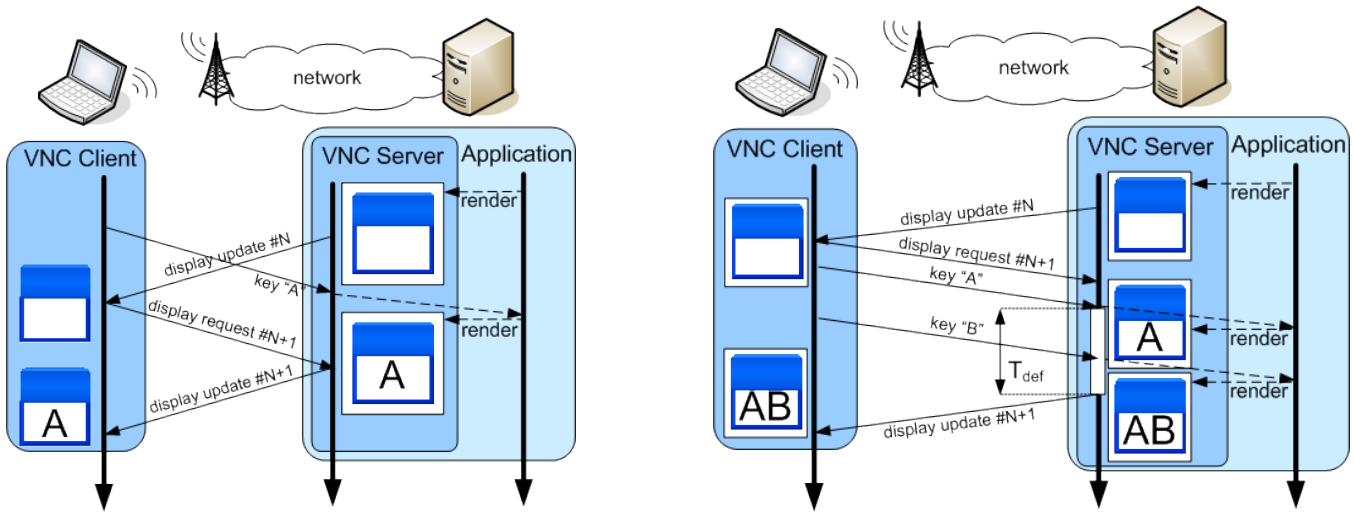
Virtual Network Computing (VNC) [19] is a remote display architecture adopting the client-pull model, that is widely used and for which an open source implementation is available. The thin client protocol that is used in the VNC architecture is the Remote FrameBuffer protocol (RFB). VNC-RFB will be used as example to detail the operational specifics of a client-pull protocol. The analysis is based on both the official VNC-RFB protocol description [20] and code analysis. The described protocol behavior was verified in the code of two VNC client implementations, RealVNC [21] (version 4.1.3 for Unix) and TightVNC [22] (version 1.3.10 for Unix).

The VNC architecture is composed of a VNC server and a VNC viewer communicating through the RFB protocol. The VNC server creates a virtual framebuffer where applications render the pixels to be displayed. The VNC viewer controls the rate of display updates by sending display update requests. In response, the VNC server encodes and transmits the regions of the virtual framebuffer that have been modified since the previous display update. While the viewer immediately issues a new request after receiving a display update, the server may not respond immediately when receiving a request. It can respond in two ways, depending on whether the virtual framebuffer was modified or not since the previous display update. Both options are depicted in Figure 2.

When a request is received, and the virtual framebuffer has been modified since the previous update was sent, the server will respond immediately to the request. Otherwise, the server will apply the *deferred update* mechanism. In this case, a timer is started at the moment new content is rendered by the application. A deferred display update is sent when this timer has expired after T_{def}. The deferred update mechanism was added to VNC because after a longer period without display changes, multiple display changes can be expected in a short interval. The deferred update mechanism should ensure that more display changes can be coalesced in a single display update. For example, the user has been reading some text, and then scrolls down to the next paragraph. Another advantage of the deferred update mechanism is that it avoids too many request-update patterns when the roundtrip time between client and server is small, e.g. when the server is deployed in the same WLAN as the thin client.

IV. Energy saving through idle time reduction

In the previous section, it was detailed how the server only sends a display update in response to a client request. As a consequence, a client must not expect a display update earlier than one network roundtrip time (RTT) after sending a request. Because of the deferred update mechanism, however, the size of the interval between sending a request and receiving a display update is variable. Currently, the WNIC stays all the time in idle mode, as it is ignorant on the arrival of the display update. It only transitions shortly to the send mode when user events need to be transmitted. In this section, the discussion will be focused on the design of an idle time reduction algorithm (ITRA) for client-pull protocols. This algorithm predicts the size of idle intervals and puts the WNIC



(a) Immediate response (b) Deferred response

Fig. 2. The VNC server can respond in two ways to a display update request. When the display has changed since the previous update was sent, a new display update is sent immediately. Otherwise, the deferred update mechanism is applied. In this case, a deferred timer is started when the display is changed the first time after receiving this request. The display update is transmitted upon expiration of the timer after T_{def}.

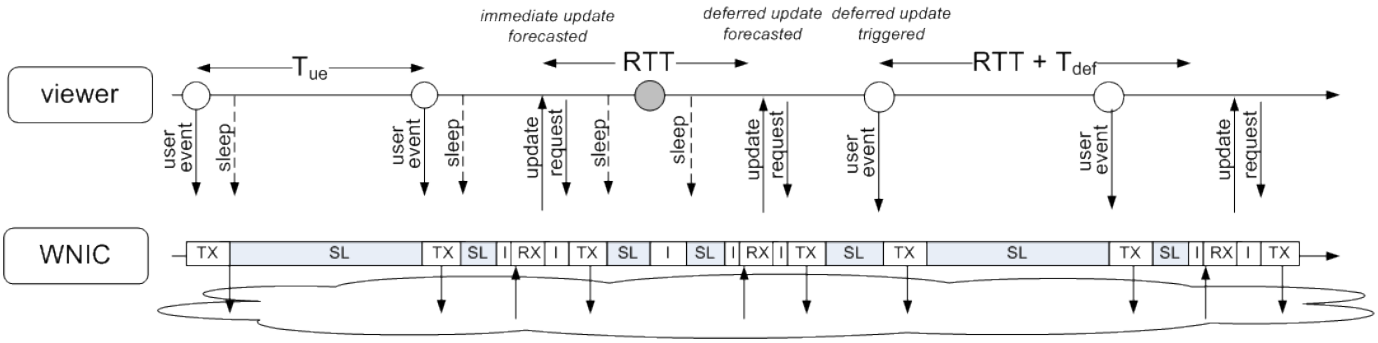


Fig. 3. The ITRA algorithm runs at the client, analyzing thin client protocol information and forecasting the arrival time of the next display update. The WNIC is transitioned to sleep mode during intervals without communication. At least every T_{ue}, a transmission opportunity for user input is scheduled. The first request is answered immediately. Because no data is transmitted at the third transmission opportunity, the virtual framebuffer at the server is not modified when the second display update request arrives. As a result, the server will answer the second request through the deferred update mechanism.

in sleep mode accordingly. In the first part of this section, the assumptions to apply ITRA are listed and explained. In the second part, the algorithm itself is introduced.

A. Assumptions

In this article, applications are targetted that only update the display after some user interaction. This includes a remote text editor, where the display changes only occur if the user types some text or uses the mouse to access a menu. Another example scenario is a user browsing to a website with static content, such as an online newspaper. In this case, the display is only updated when the user scrolls down, or clicks on a hyperlink. A similar assumption to optimize energy efficiency for web-based applications has been made before by other authors in [16]. Applications that are not covered by this assumption are mainly characterized by embedded multimedia, such as video players or websites such as YouTube. However, to display remote multimedia applications on a thin client, server-push protocols are more appropriate [17], or additional enhancements are required for client-pull protocols [23], [6].

Assuming that display updates are only caused by user input, does not imply that each user event results in a modification of the screen. Therefore, we define *effective* user events as those events that effectively result in a display update. Generally, user events are generated from the keyboard or through the pointing device (mouse). When a user strikes a key, both a key press and key release event are generated. Although two events are generated, only the key press event is an effective user event because it results in the corresponding character being drawn on the screen. Similarly, moving the mouse results in a series of position updates that are communicated to the server, but not each position update results in a display update. For example, if the user is just moving the mouse upwards to the menu bar, the screen is not updated. On the other hand, if the user has clicked on the menu bar, a drop-down menu is shown. When the user moves the mouse to hover over the menu, the display needs to be updated to highlight the menu

item at the current position. Because of this ambiguity, it was decided to classify each mouse event as an effective event. This design decision might limit the resulting energy gains because the algorithm erroneously expects display updates. However, this ensures that no display updates will be missed because the WNIC is in sleep mode. It is important to receive all display updates, as users are very sensitive to application reactivity delay when using the mouse.

Furthermore, the algorithm assumes that a single thin client protocol connection is established between client and server. This avoids putting the WNIC in sleep mode based on traffic analysis from one connection, while data may be sent to the device over another connection. This assumption can be advocated for the thin client case, where only a single network connection is required for the thin client protocol running between the thin client and the remote virtualized desktop environment [24]. We want to stress that this assumption does not a priori exclude the possibility of applying ITRA in case of users running multiple applications at the same time. Nevertheless, to support this case, ITRA's decision logic must be extended to cope with the more complex display update patterns. For example, the position of a mouse click determines to which application the mouse click pertains. It can be expected that display updates will arrive more frequently when multiple applications are visible. As a result, potential sleeping intervals will be shortened.

Lastly, user events are often generated in an unpredictable way and shortly after each other, e.g. when a series of pointer position updates must be communicated to the server. Immediately transmitting each user event limits the size of potential sleep intervals, because the WNIC must be frequently transitioned to send mode to transmit small chunks of data. Therefore, a user event buffer is implemented at the thin client protocol layer. During an interval of T_{ue}, user events are accumulated in the buffer and the WNIC can stay in sleep mode if no display update is expected. At the end of the interval, a transmission opportunity is scheduled and all buffered user events are handed over at once to the WNIC for transmission. Buffering user events increases the delay between the user event and the result being presented on the screen. As a consequence, the period T_{ue} of user event transmission opportunities should be carefully chosen.

B. Algorithmic details of ITRA

The energy optimizing cross-layer approach is presented in Figure 3. As explained in the previous section, user event transmission opportunities are scheduled with a period of T_{ue}. At these moments, all buffered user events are handed over from the thin client protocol layer to the lower layers of the protocol stack. In between two transmission opportunities, the WNIC is put into sleep mode. It only needs to wake up to receive a display update and immediately transmit the subsequent display update request. After each transmission of a user event or display update request, the client applies ITRA to determine the size Z of the next sleep interval. The WNIC is transitioned to sleep mode either until the next user event transmission opportunity, or until the arrival of the next display update. The delay to enter or exit the sleep mode is in the sub-microsecond order [25]. Compared with the server response and network propagation delays, which are both in the order of tens of microseconds, the transition delay can be neglected.

A pseudo code overview of ITRA is presented in Listing 1, the symbols are explained in Table I. ITRA is invoked each time one of the following three events has occurred: the client has transmitted a display update request, the client has received a display update, or a user event transmission opportunity is scheduled. In the first part of the algorithm, thin client protocol information is analyzed and the necessary state variables are updated. In the second part of the algorithm, the appropriate WNIC sleep time is calculated.

When a display update is received, the ITRA state changes to S_{noreq} until the next display update request is transmitted. Furthermore, the variable D is decremented. The usage of this variable will be described further. Lastly, the variable N_{disp} is updated to contain the sequence number N_{disp} of the last user event of which the result is contained in the received display update. This information is available at the client through a slight modification of the VNC-RFB protocol. When generating a display update, the server adds 2 bytes containing the sequence number of the last user event received at that moment.

When a display update request is transmitted, ITRA needs to determine whether the request will be answered immediately or through the deferred update mechanism. An immediate display update is only to be expected if one or more effective user events that have been transmitted to the server, were not yet contained in the last received display update. To this end, ITRA compares N_{disp} with the sequence number of the last effective user event that was transmitted (N_{eff}). If there are any user events that were transmitted but not yet displayed, the next (immediate) update is expected to arrive over 1 RTT. Otherwise, ITRA enters the $S_{def\_pending}$ state, waiting until the deferred update timer is triggered by the transmission of an effective user event. This state is changed to S_{def} when a key press or mouse event is transmitted during a subsequent transmission opportunity. As explained in Figure 2(b), the deferred display update is to be expected over 1 RTT + T_{def}.

After each user event transmission opportunity, regardless whether actual data was transmitted or not, ITRA checks if the delay of the currently expected display update does not increase a configurable threshold ERR_{max}. A late arrival of a display update can have two causes: either ITRA has erroneously regarded a user event as effective (and hence wrongly predicted the arrival time), or the display update was lost, due to the WNIC being in sleep mode or due to packet loss in the network. Because each key press certainly results in a display update, the first cause can only occur for mouse events, e.g. when the user is moving the mouse upwards to the menu bar, but no display updates occur. Therefore, when a late display arrival occurs, ITRA's reaction will depend on whether the user is currently handling the mouse or the keyboard, which is reflectedin the variable $mouseActivity$. When a late display update arrival occurs and the user is currently handling the mouse, ITRA will assume that the update is triggered by a later user event transmission opportunity than the one that

was currently used to estimate the arrival time, and revise the expected arrival time accordingly. To this end, an ordered list L_{opp} is maintained, containing the sequence numbers of all previous transmission opportunities at which actual user event data was transmitted. The next transmission opportunity is retrieved from the list and used to revise the arrival time of the display update. If no new transmission opportunity can be found in the list, the state is changed to $S_{def\_pending}$.

In the other case, when an update is late and the user is currently generating input by the keyboard, we assume the late arrival is due to packet loss, either because it was lost in the network or because the WNIC was in sleep mode. Because the underlying TCP protocol will reduce its send window and throughput, this data loss will result in an increased value of the propagation delay between client and server that is observed at the application layer. In turn, this might lead to wrong ITRA predictions and additional data losses because the WNIC is erroneously put in sleep mode. To avoid that new data is lost and hence to enable TCP to return to the steady state, the WNIC will not be put in sleep mode until $Q_{disable}$ more display updates have been arrived. Because the interval between two subsequent display updates is at least 1 RTT, ITRA will not send any sleep instructions to the WNIC for a period of at least $Q_{disable}\cdot$ RTT. The parameter $Q_{disable}$ can be adapted to the amount of congestion or interference that is observed on the network. Clearly, this design choice prioritizes the user perceived responsiveness and may affect to the energy efficiency. However, we motivate this choice by the fact that the application reactivity delay requires particular attention in the thin client paradigm, as it is already inherently bound by the roundtrip time between client and server.

In the last part of the algorithm, the actual sleep time for the WNIC is calculated. The WNIC will either be put in sleep mode until the next user event transmission opportunity, or until the expected arrival time of the next display update. However, the WNIC is not instructed to enter the sleep mode after receiving a display update, because the subsequent request will be transmitted immediately.

C. 802.11 Power Saving Management (PSM)

To minimize the power consumption of 802.11, the Power Saving Management algorithm was added to the standard [26]. At regular intervals, usually every 100 ms, the access point broadcasts a beacon, indicating a.o. for which mobile hosts the access point has at least one frame buffered. It is however generally acknowledged that PSM is not suited for light traffic load or latency-sensitive applications [27], [10], [28]. Consequently, we therefore assume to PSM is disabled when ITRA is applied at the client.

Another consideration is that ITRA might put the WNIC in sleep mode when the access point broadcasts its beacons. Missing beacons poses however no fundamental problem, as the information in the beacon is only required during the association process of the mobile host to the access point. Previously, other authors have already proposed energy saving schemes in which several beacons are not received [29], [30], [31].

Listing 1 Idle Time Reduction Algorithm (ITRA)

Precondition: RTT, T_{ue}, T_{def} are initialized
Precondition: This algorithm is called after an event E

```
/* Part A: Thin client protocol analysis */
switch (E) {
    case E_DU : /* display update received */
        S == S_noreq
        N_disp = E.disp
        D = (D > 0) ? (D − 1) : 0
        break
    case E_REQ : /* display update request transmitted */
        if N_disp < N_eff
            S = S_imm
            T_DU = CT + RTT
            N_trigger = N_eff
        else
            S = S_def_pending
        break
    case E_opp : /* user event transmission opportunity */
        N_opp + +
        T_opp = CT + T_ue
        if (E.tx_keypress || E.tx_mouse) /* effective UE */
            N_eff = N_UE
            L_opp ← N_opp
            if S == S_def_pending
                S = S_def
                T_DU = CT + RTT + T_def
                N_trigger = N_opp
        /* Take appropriate action when update is late */
        if (CT - T_DU) ≥ ERR_max
            if mouseActivity
                N_new_trigger ← L_opp
                if !N_new_trigger
                    S = S_def_pending
                else
                    T_DU = (N_new_trigger − N_trigger) · T_ue
                    N_trigger = N_new_trigger
            else
                D = Q_disable
        break
}

/* Part B: sleep time determination */
if D == 0
    if S == S_def_pending
        Z = T_ue
    else if S == S_imm||S == S_def
        Z = min(CT − T_DU,CT − T_opp)
else
    Z = 0 /* sleeping is disabled */
```

TABLE I
EXPLANATION OF THE SYMBOLS USED IN THE DESCRIPTION OF THE ITRA ALGORITHM

symbol	description
CT	current time
D	boolean indicating whether WNIC sleep mode instructions are disabled
E_{DU}	display update received
E_{REQ}	display update request transmitted
E_{opp}	a user event transmission opportunity occurred
$\mathrm{ERR}_{\mathrm{MAX}}$	maximum tolerated error on latency prediction
L_{opp}	ordered list of sequence numbers of transmission opportunities with effective user events
N_{disp}	sequence number of last displayed user event
N_{eff}	sequence number of last effective user event
N_{opp}	sequence number of last user event transmission opportunity
$N_{trigger}$	sequence number of transmission opportunity that triggered the deferred update
N_{UE}	sequence number of last user event
$Q_{disable}$	amount of display updates to be received before WNIC sleep mode instructions are reactivated
S	current status of the ITRA algorithm
S_{noreq}	currently no unanswered display request has been transmitted to the server
$S_{def\_pending}$	a deferred update is expected, but the deferred timer is not yet activated
S_{def}	a deferred update is expected and the deferred timer is activated
S_{imm}	an immediate update is expected
T_{DU}	expected arrival time of next display update
T_{ue}	time between two subsequent user event transmission opportunities
T_{opp}	time of next user event transmission opportunity
W	constant indicating how many display updates must be received before retaking the algorithm
Z	instructed WNIC sleep time

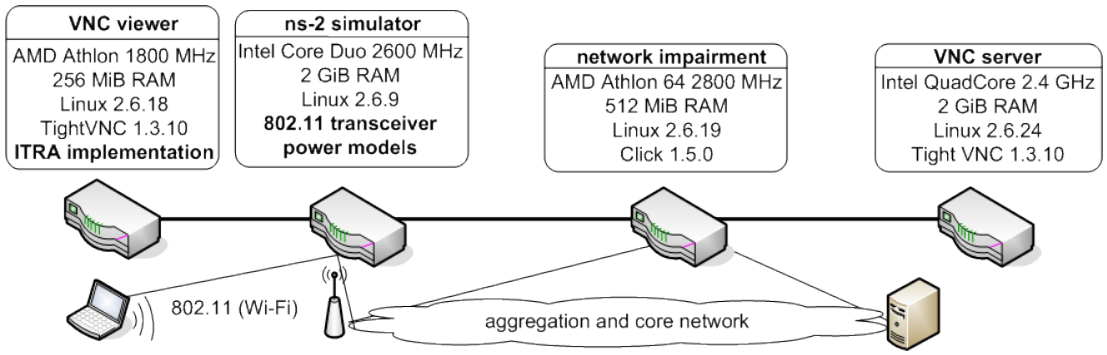


Fig. 4. Test set-up comprising a VNC viewer and server, a ns-2 node and an additional impairment node.

V. SIMULATION RESULTS AND DISCUSSION

The first part of this section elaborates on implementation details of ITRA, as well as on the test set-up that was used for the validation experiments. In the second part, ITRA is validated by automatically generating user events at a fixed rate. In the last part, validation results are presented that were obtained through actual user input traces.

A. Test set-up

ITRA was implemented in TightVNC version 1.3.10 for Linux. Although different VNC alternatives exist, TightVNC is one of the most bandwidth efficient VNC clients because of a better compression of display updates [23].

As explained in section I, we adopt the scenario of a wireless thin client that connects over a Wi-Fi link to the application server. The total roundtrip time (RTT) between client and server is composed of the wireless link latency between client and access point (RTT_{CA}), and the latency of the aggregation and core network between access point and server (RTT_{AS}). If the application server is located in the same WLAN as the client, $RTT_{AS} = 0$ and $RTT = RTT_{CA}$.

The value of the parameter $Q_{disable}$ of the ITRA algorithm was set to 2. This value was empirically determined and provided good results.

The testbed is shown in Figure 4. It is composed of a client and server machine that are interconnected by an impairment node and a ns-2 node, acting both as 802.11 wireless tranceiver (MAC+PHY) and channel simulator.

The impairment node is a Click modular router [32] and is used to configure RTT_{AS}. The Click framework contains a wide range of packet processing modules that can be combined in a flexible way to describe a forwarding packet path. Besides traditional packet storage, scheduling and forwarding functionality, also traffic shaping and network impairment modules are available. In our simulations, we have used the *DelayShaper* module to introduce additional latency RTT_{AS} on the end-to-end path. This latency was equally split over both directions. For example, to configure RTT_{AS} = 100 ms, the DelayShaper elements in up- and downlink were each configured to delay the packets with 50 ms.

The ns-2 emulation framework simulates the WNIC of the thin client, the IEEE 802.11 access point and the Wi-Fi channel in between. The framework is set up in emulation mode,

taking the application packets from the network and, after being passed through the emulated Wi-Fi channel, delivers these packets again to the network in real time. The communication with the ns-2 machine is established over TAP/TUN interfaces that are completely transparent to the VNC client and server components. Wireless channel parameters such as the average path loss, coherence time or background traffic are configured through a Tool Command Language (TCL) script. Besides emulating the Wi-Fi channel, the ns-2 framework also measures the WNIC energy consumption, as it was provided with the power models of a platform for software defined radios (SDR) [33]. These power models are based on the estimation of the gate level activity of the platform. Monitoring specific parameters such as modulation, code rate, radio mode, power amplifier settings, and the time spent in each mode, the energy consumed by the whole platform is estimated based on the energy consumed by each of its components. The energy efficiency of the platform is further enhanced by a MAC/PHY cross layer control algorithm. Contrary to conventional MAC control algorithms that try to minimize the energy by transmitting the data as fast as possible, the most energy-efficient modulation is chosen to transmit the data, given the existing path loss and data rate requirements. To this end, the simulator is equipped with a pre-computed database, containing the optimal set points for each given combination of path loss and required goodput (error-free throughput). More details are available in [34]. In the remainder of this paper, this MAC/PHY cross layer approach will be used as reference solution (RS) to benchmark the energy efficiency of the ITRA algorithm.

B. Validation with automated user event generation

The results presented in this section are obtained by automatically generating user events at a fixed rate. In the experiments, keystrokes were generated during 2 minutes. The interval between subsequent user events was equal to the period of user event transmission opportunities T_{ue}. Alternating with a fixed period of T_{ue}, a key press or a key release event was generated. As a result, at every transmission opportunity there was exactly 1 user event to be transmitted. Dynamic user event rates are discussed in section V-C.

In the experiments, the text editor Open Office Writer 3.0 was used. This application is covered by the assumptions made in section IV-A. Although alternating key press and key release events were generated, only key press events effectively result in a modification of the display, i.e. showing the character. Therefore, we introduce $T_{ue,eff} = 2 \times T_{ue}$, to denote the interval between subsequent effective user events. As will be explained in the next section, $T_{ue,eff}$ is a relevant parameter to interpret the results correctly.

The efficiency of ITRA was investigated for a wide range of network conditions. The aggregation and core network latency (RTT_{AS}) was varied between 0 and 100 ms, whereas the average path loss of the wireless channel (PL) was varied between 60 and 90 dB. The interval between two user event transmission opportunities (T_{ue}) was chosen between 20 and 100 ms. In the remainder of this section, the influence of each parameter is discussed separately. Lastly, the reader should be notified that $T_{def} = 40$ ms in VNC.

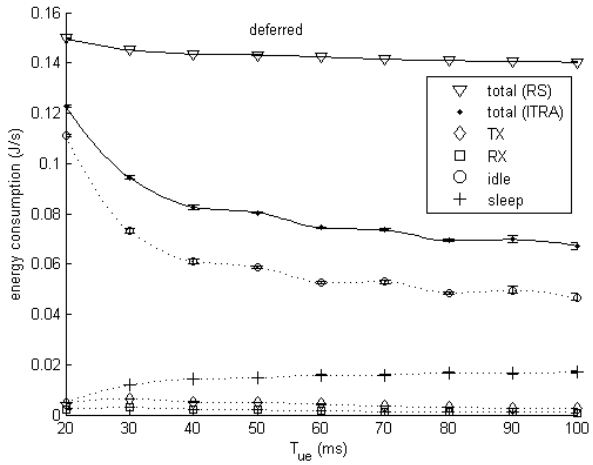
1) Influence of RTT/T_{ue} ratio: As shown in Figure 2, the values of RTT and T_{ue} determine the rate of display updates received by the client. Hence, these parameters have a direct influence on the total energy consumption. In general, a lower total energy consumption is expected for higher values of RTT and T_{ue}, because fewer data will be transmitted and received by the WNIC. Indeed, increasing values of T_{ue} mean user input is transmitted less frequently, and fewer display updates are received. Hence, less energy is consumed in the send and receive mode, and, more importantly, longer WNIC sleep intervals can be identified. Similarly, for increasing values of RTT, less display updates per second are received because of the client-pull operation of VNC.

Besides the individual value of RTT and T_{ue}, also the ratio of these two parameters is important, as it determines whether a display update request will be answered immediately or through the deferred update mechanism. In particular, the value of $T_{ue,eff}$ as compared to RTT is important. Based on the ratio of these two parameters, and referring to Figure 2, three possible protocol regimes can be distinguished. If $T_{ue,eff} < RTT$, all display update requests will be answered immediately. In this case, the rate of effective user events is very high and the virtual framebuffer is always modified when a new display update request arrives. In contrast, if $T_{ue,eff} > RTT + T_{def}$, the rate of user events is very low and each user event arriving at the server triggers a deferred display update. If $RTT < T_{ue,eff} < RTT + T_{def}$, both immediate and deferred updates can be expected. It is clear that in the latter case, the prediction of the arrival time of the next display update is more complicated.

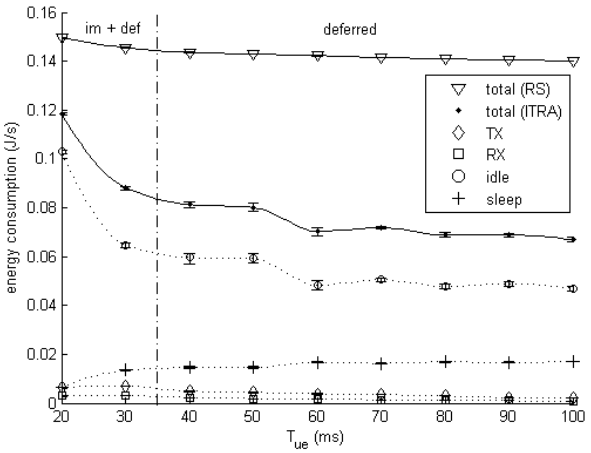
Figure 5 shows the reduction in average energy consumption per second (J/s) achieved by ITRA compared to RS, for 4 different values of RTT_{AS}. Furthermore, the energy consumed in each of the 4 WNIC modes (send, receive, idle and sleep) is shown when ITRA is applied. Because the results on average energy consumption are directly related to the used hardware, we have provided figures on the time spent in each state in Table II. These figures should reflect the efficiency of ITRA in a more hardware-independent way.

For RS, the total energy consumption decreases as expected for higher values of T_{ue} and RTT. The same general trend is observed when ITRA is applied. For ITRA, however, additional observations must be made. To interpret the results correctly, the analysis must be carried out based on the $RTT - T_{ue,eff}$ ratio and the three regimes that were described above. These regimes are indicated on Figure 5 by vertical delimiters.

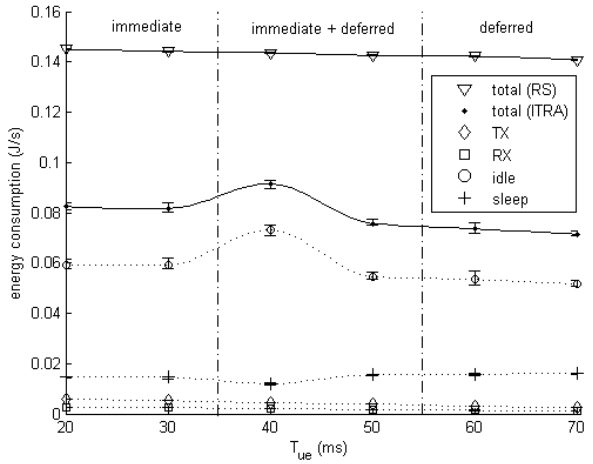
The general decreasing trend for higher values of T_{ue} is slightly interrupted in the regime with both immediate and deferred updates. In this regime, the server will sometimes send a deferred update whereas ITRA expected an immediate update or vice versa. Possible causes for these variations are network jitter and server background processes, affecting RTT and the server calculation time of a display update. This will result in display updates arriving too early or too late, as compared to ITRA's prediction. As explained in section IV-B,



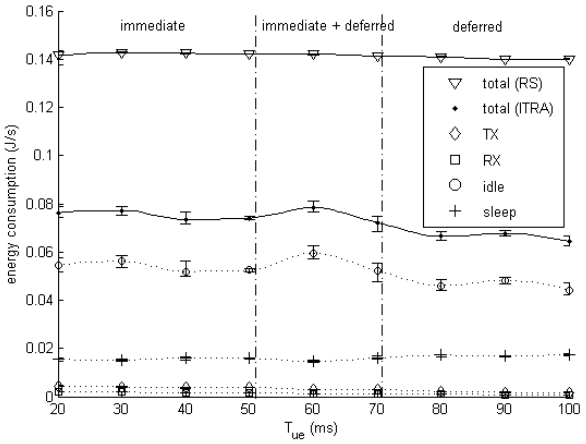
(a) RTT_{AS} = 0 ms



(b) RTT_{AS} = 30 ms



(c) RTT_{AS} = 70 ms



(d) RTT_{AS} = 100 ms

Fig. 5. Comparison of the average energy consumption for RS and ITRA. The value ranges are indicated as well. Furthermore, for ITRA, the energy consumed in send (TX), receive (RX), idle and sleep mode is shown. The vertical lines delimit the different display update regimes (immediate or deferred update), with T_{def} = 40 ms for VNC. The results shown are for a $PL = 60$ dB and no background traffic on the wireless link.

ITRA will react to these late arrivals by not putting the WNIC in sleep mode until two more display updates are received. Although variations in arrival time also occur in the other two regimes, their effect is too small to trigger a deferred update when an immediate update was expected or vice versa because the value of $T_{ue,eff}$ is too small or too large. Hence, ITRA is less frequently disabled. The performance of ITRA could potentially be improved if TCP layer monitoring information is included in the framework. This information would allow to identify the exact cause of a late display update. If the display update is late because of network jitter or server background processes, no retransmissions at the TCP layer would be observed and ITRA should not be disabled. If a display update was missed because the WNIC was erroneously instructed to enter the sleep mode, TCP retransmissions would occur.

On Figure 5, it can observed that the energy gain is slightly more pronounced for specific combinations of T_{ue} and RTT, e.g. for $(RTT_{AS}, T_{ue}) = (30, 60)$ and $(RTT_{AS}, T_{ue}) = (100, 80)$. In these cases, the arrival of a display update often nearly coincides with a transmission opportunity. Consequently, the WNIC does not need to wake up between two user event opportunities and can stay in sleep mode for longer intervals. For example, in the case of $(RTT_{AS}, T_{ue}) = (30, 60)$, every key press event triggers the deferred update mechanism, and the update is received approximately $60 + 40$ ms after sending the event to the server. At the same moment, the client will transmit a key release during a transmission opportunity (note that in the meantime, a transmission opportunity occurred during which a key release event was transmitted).

2) Influence of path loss: The previous section mainly focused on the latency between access point and server (RTT_{AS}). In this section, this parameter is kept constant, and the impact of varying channel conditions of the Wi-Fi link between terminal and access point is investigated by experiments for different values of the channel path loss. Variations in distance between client and access point, indoor walls that obstruct the line-of-sight communication or interference and

TABLE II
COMPARISON OF THE TIME SPENT IN EACH WNIC MODE WITH AND WITHOUT APPLYING ITRA. THE RELATIVE REDUCTION IN IDLE TIME AND TOTAL ENERGY CONSUMPTION ARE INDICATED. THE MEASUREMENT VALUES WERE OBTAINED WITH $PL = 60$ DB. THE VALUES OF RTT_{AS} AND T_{ue} ARE IN MS.

Variables		Avg. time per second [ms]								Reduction[%]	
		no ITRA				ITRA				idle	total
RTT_{AS}	T_{ue}	send	recv	idle	sleep	send	recv	idle	sleep	time	energy
0	20	7	14	971	8	3	6	809	182	16.7	18.0
	50	3	6	988	3	3	5	428	564	56.7	43.8
	100	2	3	994	2	1	3	340	656	65.8	52.1
30	20	6	13	975	6	4	8	748	240	23.3	20.9
	50	3	5	990	3	2	5	426	567	57.0	44.3
	100	1	3	995	1	1	2	341	656	65.7	52.2
50	20	5	10	980	5	5	10	464	522	52.7	39.9
	50	2	5	990	3	2	5	437	557	55.9	44.1
	100	1	3	994	2	1	3	345	651	65.3	51.5
70	20	4	8	984	4	4	7	432	557	56.1	43.2
	50	2	5	990	3	2	5	396	597	60.0	46.9
	100	1	2	995	1	1	3	347	649	65.1	51.2
100	20	4	6	986	4	3	6	395	596	59.8	46.9
	50	2	5	990	2	2	5	382	611	61.4	47.9
	100	1	2	995	1	1	2	322	675	67.6	53.9

fading effects in public places primarily result in varying values of the Wi-Fi channel path loss. For the experiments reported in this section, the ns-2 simulator was configured with average path loss values between 60 dB and 90 dB. On top of this average value, additive Gaussion noise was added.

Figure 6 shows the energy consumed and the time spent in each of the 4 possible WNIC modes. The presented results cover all three relevant $T_{ue} - RTT_{AS}$ ratios, as explained in previous section. For the simulated scenarios, the value of the path loss value does not significantly influence the total energy consumption nor the energy gain that is realized by applying ITRA. For example, for $T_{ue} = 50$ ms, the relative energy reduction decreases from 53.9 % for $PL = 60$ dB to 48.4 % for $PL = 90$ dB, and the relative idle time reduction decreases from 67.6 % to 61.2 %. With higher values of the channel path loss, more MAC/PHY retransmissions are required. This results in more late display update arrivals, and ITRA is more frequently disabled. However, as can be seen in Figure 6, this effect seems only relevant for PL = 90 dB. For lower PL values, the efficiency of ITRA remains constant.

The minor influence of the average channel path loss can be explained by the limited bandwidth consumption required for the user events and the display updates, as compared to the available channel bandwidth. Table III provides values of the total VNC bandwidth consumption.

TABLE III
UP- AND DOWNSTREAM BANDWIDTH CONSUMPTION OF VNC-RFB PROTOCOL FOR THE SCENARIO OF SECTION V-B. THE VALUES OF RTT_{AS} AND T_{ue} ARE IN MS.

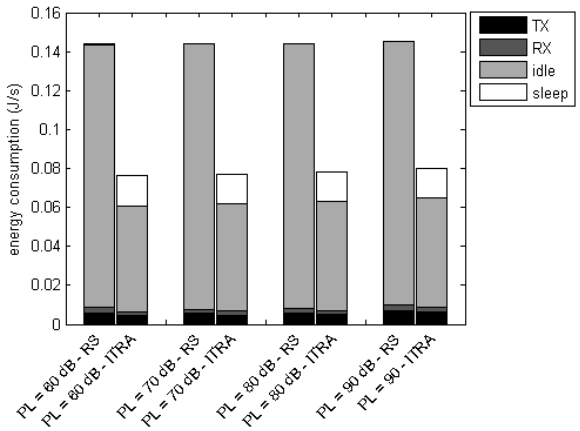
		bandwidth [kbps]	
RTT_{AS}	T_{ue}	up	down
0	20	44.25	596.77
	50	17.21	227.71
	100	8.96	106.97
50	20	40.52	447.60
	50	17.23	228.34
	100	8.96	109.71
100	20	35.23	246.72
	50	17.17	225.03
	100	8.96	106.70

These values are well below the (theoretical) maximum Wi-Fi throughput. When additional background traffic is present on the channel, e.g. to other devices connected to the same access point, the bandwidth available for the thin client connection is reduced. Hence, a bigger impact on the energy consumption can be expected. This is studied in the next section.

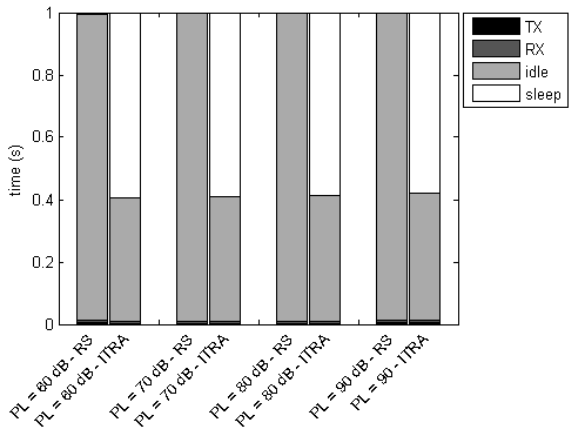
3) Influence of congestion: When multiple terminals are communicating with the access point, the available bandwidth per terminal is reduced. In this paragraph, all traffic between other terminals and the access point, is referred to as background traffic (BT). To investigate the influence of this background traffic, the ns-2 node in the testbed has been configured with BT = 0, 1, 6 and 12 Mbps while performing the same automated text editing scenario as in the previous two sections. Figure 7 shows the energy consumed and the time spent in each of the 4 possible WNIC modes. Similar to the previous section, results are presented for $T_{ue} = 20, 50, 100\ ms$ in order to cover the three relevant $RTT_{AS} - T_{ue}$ ratios, as explained in section V-B1.

For RS, higher levels of background traffic result in higher total energy consumption. For example, in case $T_{ue} = 20$ ms, the average energy consumption increases by 40.3 % when a background traffic of 12 Mbps is present on the wireless channel. The increase of total energy consumption is primarily due to an increase of the energy consumed in the receive state. Because the wireless channel is a shared medium, the WNIC senses all frames on the channel and needs to decode at least the MAC address in the frame header to determine if the frame is destined for this device.

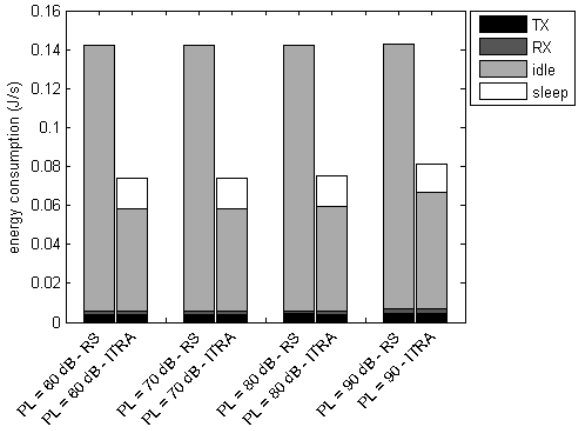
In the presence of background traffic, the performance of ITRA decreases. For example, ITRA reduces the idle time by 59.8 % for BT = 0 Mbps and T_{ue} = 20 ms, whereas this decreases to 20.5 % for BT = 12 Mbps. This is reflected by the energy gains achieved through ITRA: 46.9 % for BT = 0 Mbps, and by 17.5 % for BT = 12 Mbps. The background traffic results in more collisions and packet loss at the wireless channel. Both effects delay the arrival of display updates due to retransmissions at the MAC and TCP layer. Analysis of the



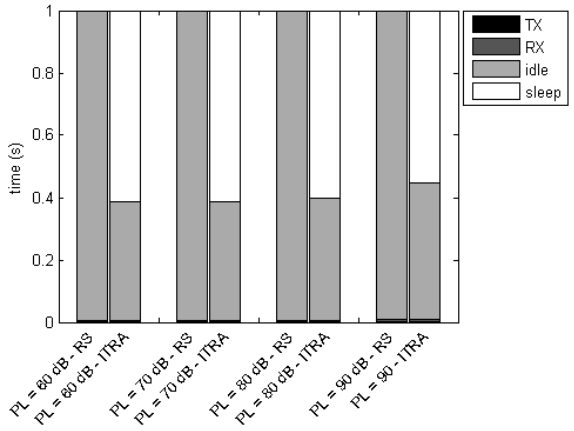
(a) Energy consumption (T_{ue} = 20 ms)



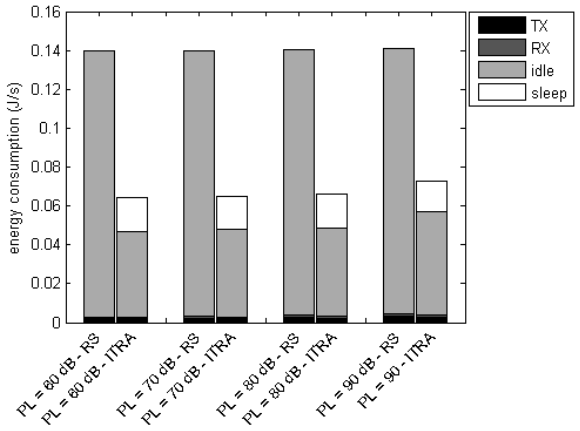
(b) Time distribution (T_{ue} = 20 ms)



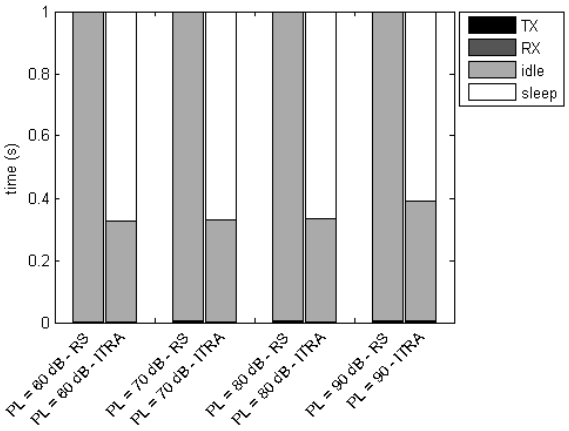
(c) Energy consumption (T_{ue} = 50 ms)



(d) Time distribution (T_{ue} = 50 ms)

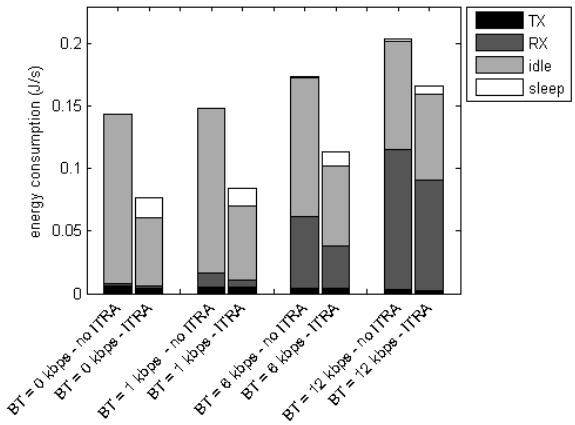


(e) Energy consumption (T_{ue} = 100 ms)

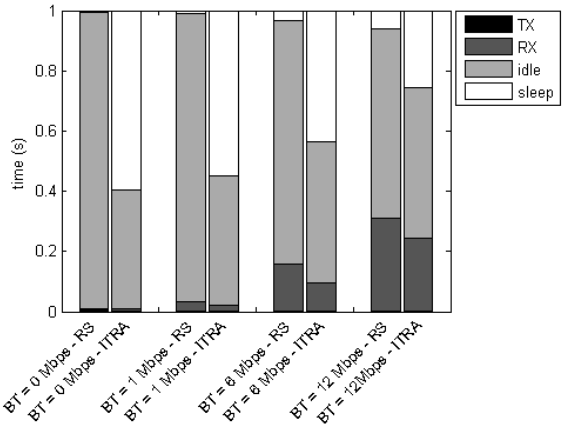


(f) Time distribution (T_{ue} = 100 ms)

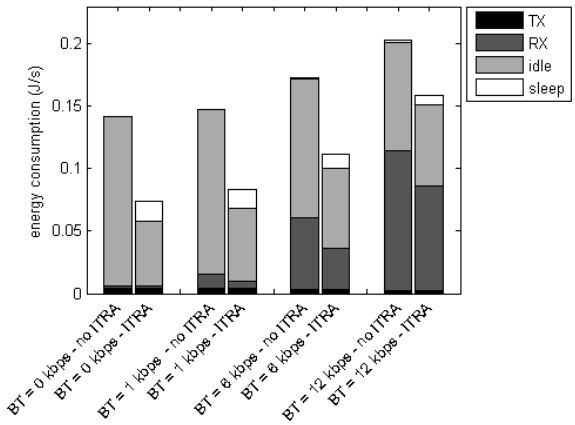
Fig. 6. Influence of path loss on WNIC energy consumption and time spent in each state. Higher path loss values result in a slight increase of the total energy consumption, but do not degrade the effectiveness of ITRA.



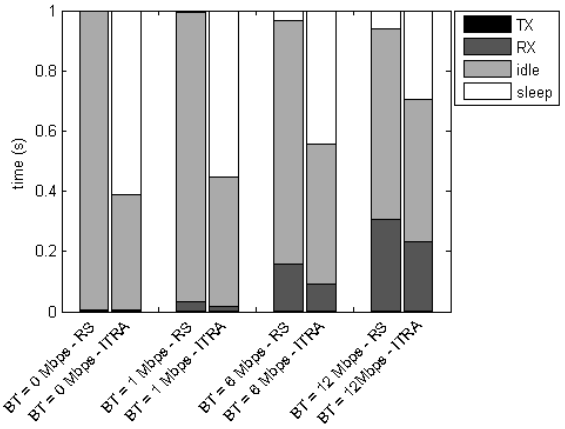
(a) Energy consumption (T_{ue} = 20 ms)



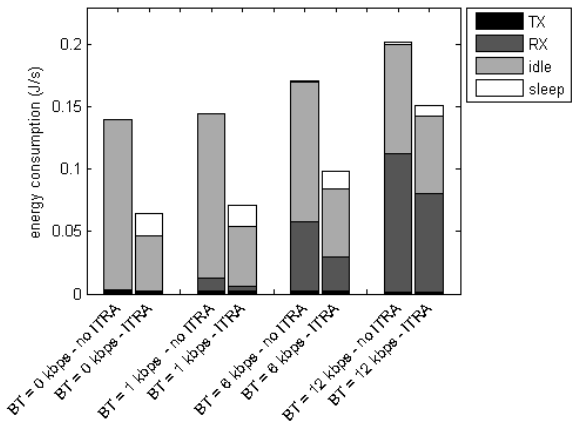
(b) Time distribution (T_{ue} = 20 ms)



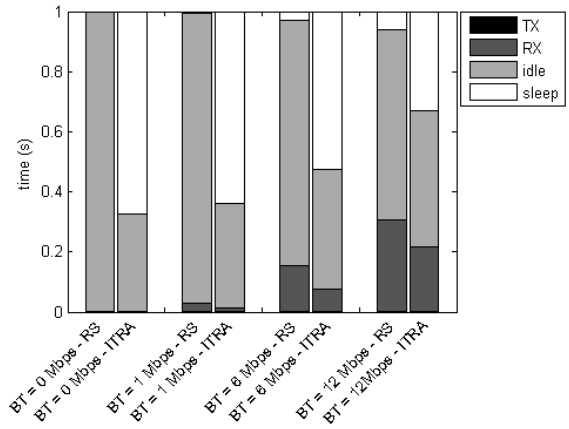
(c) Energy consumption (T_{ue} = 50 ms)



(d) Time distribution (T_{ue} = 50 ms)



(e) Energy consumption (T_{ue} = 100 ms)



(f) Time distribution (T_{ue} = 100 ms)

Fig. 7. Influence of background traffic on WNIC energy consumption and time spent in each state, with $PL = 60$ dB and $RTT_{AS} = 100$ ms. Higher levels of background traffic result in an increase of the time spent in the receive state.

TABLE IV
ITRA ENERGY EFFICIENCY GAINS FOR ACTUAL USER TRACES. EACH USER HAS A DIFFERENT NUMBER OF KEY PRESSES DUE TO THE CORRECTION OF TYPING ERRORS. THE RESULTS WERE OBTAINED FOR PL = 60 DB AND RTT_{AS} = 100 MS. THE INDICATED ENERGY REDUCTIONS ARE AVERAGED OVER 10 ITERATIONS.. THE ACCURACY RESULTS REPRESENT THE 90 % PERCENTILE FOR T_{ue} = 50 MS.

user	trace length [s]	# user events key press	# user events mouse	ITRA energy reduction [%] 20 ms	50 ms	100 ms	prediction error [ms]
0	193	951	357	31.20	34.14	31.02	6.22
1	255	935	354	29.96	31.24	30.33	8.47
2	236	943	301	32.99	36.40	31.91	6.26
3	262	1001	275	32.55	32.75	31.79	5.71
4	121	899	338	31.41	32.88	29.01	6.19
5	177	984	244	32.98	33.64	29.07	6.36
6	236	1044	385	30.07	31.43	28.73	6.55
7	230	982	269	30.99	34.23	29.92	6.36
8	226	1068	485	34.37	36.82	32.73	6.28
9	257	956	381	29.86	31.20	28.87	6.54

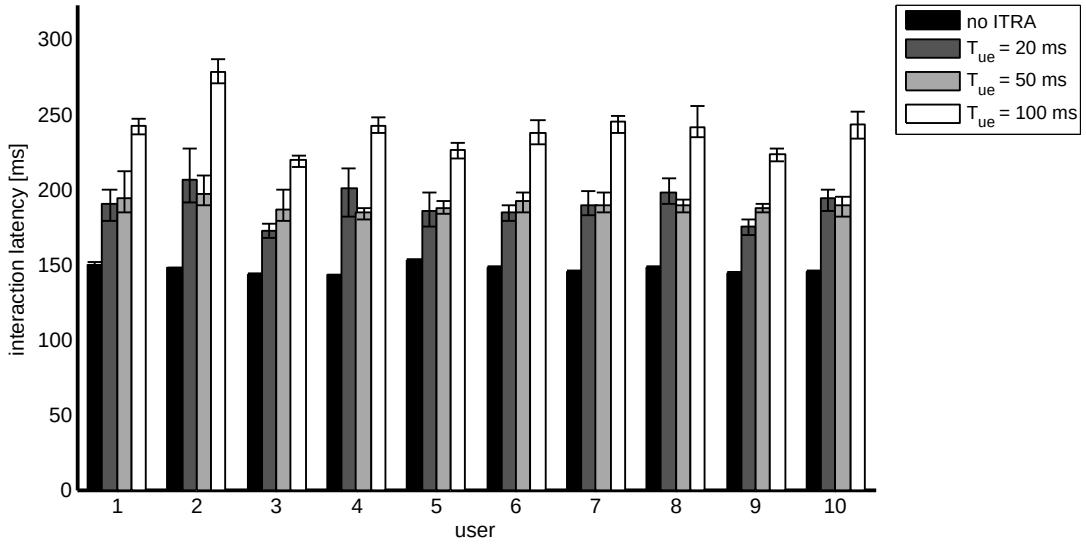


Fig. 8. Influence of applying ITRA with different values of T_{ue} on the average interaction latency. For each user, 10 iterations were run and the minimum and maximum measured value are indicated.

experimental results showed that these effects are the main cause of retransmissions, and that only few retransmissions are due to the WNIC being in sleep mode when data is transmitted to it. The additional reduction of the energy gain due to ITRA being temporarily disabled after a late display update is only minor. In additional experiments with a modified version of ITRA that does not disable the algorithm after a late display update, the energy reduction was increased by less than 1 %.

C. Validation with actual user traces

In the previous section, keystrokes were automatically generated at a fixed rate. This allowed to study the influence of various network conditions on the performance of ITRA. In order to assess the ITRA energy gains with varying user event rates, experiments were performed with actual user input traces. To this end, 10 persons were asked to execute a sample text editing scenario in OpenOffice Writer. Each user first transcribed a paragraph of 339 characters and inserted an image by navigating with the mouse to the appropriate menu item. After inserting the image, another paragraph of 540 characters was transscribed. The sample scenario was specifically chosen to include both key and mouse events. All user input was captured, resulting in 10 traces with realistic user input sequences. Afterwards, the user input traces were exactly replayed with the interval T_{ue} set to 20, 50 and 100 ms.

Table IV presents details of each user trace, as well as the relative energy gains and the accuracy of ITRA predictions.

For the simulated scenarios, relative energy gains between 28.73 % and 36.4 % are achieved. In general, the highest relative energy gains are realized for $T_{ue} = 50$ ms. This can be attributed to the fact that longer WNIC sleep intervals are realized because user event transmission opportunities coincide with the arrival of a display update. The same effect was observed in section V-B. The results indicate that by appropriately tuning the value of T_{ue} to the network roundtrip time RTT, an additional energy gain of 1-3 % can be achieved. However, the value of T_{ue} also affects the interaction latency that is perceived by the user, e.g. between generating a key stroke and the character being presented on the screen. Larger values of T_{ue} will result in longer buffering times. Therefore, it is important to quantify the increase of the interaction latency by applying ITRA. For each of the 10 user traces and RTT_{AS} = 100 ms, Figure 8 compares the average interaction latency that is experienced without ITRA with the

latency for T_{ue} = 20, 50 and 100 ms. Although generally the interaction latency increases with higher values of T_{ue}, the relation is not linear, as can be seen on Figure 8 by comparing the results of T_{ue} = 20 ms and T_{ue} = 50 ms. Furthermore, the optimal value of Tue, in terms of interaction latency depends on the user at hand, i.e. his typing speed. Whereas for some users T_{ue} = 20 ms results in less interaction latency, other users may benefit more from T_{ue} = 50 ms. These variable effects can be attributed to the deferred update mechanism. Dependent on the rate of user events, Tue can under specific circumstances be increased without degrading the interaction latency. In a previous publication [35], we have extensively studied this effect and have developed models to determine the most appropriate value of T_{ue}. It can be concluded that the value of T_{ue} needs to be adapted to the specific context of network status and the user.

Furthermore, Table IV contains the 90 % percentile of the prediction errors obtained for T_{ue} = 50 ms. For each user, this percentile is approximately 6 ms. Deeper analysis indicated that this error is due to the time required at the server to calculate and transmit a display update, whereas our model assumes a zero calculation time. An additional parameter should be added to the ITRA model to account for the finite calculation time. However, the specific value of this parameter would have to be benchmarked for each individual server. Furthermore, server calculation times can be minimized by deploying faster hardware.

VI. Conclusions

In the thin client paradigm, the exection of application logic is outsourced to a remote server. The exchange of thin client protocol data with the server results in a significant increase of the WNIC power consumption. Especially for mobile thin clients equipped with limited battery capacity, energy optimization schemes are required.

The algorithm presented in this paper aims to transition the WNIC to the energy-conserving sleep mode during intervals when no display updates from the server are expected. To this end, the algorithm analyzes the transmitted user events and display update requests to predict the arrival of the next display update. In between two user event transmission opportunities, the WNIC is transitioned to the energy-conserving sleep mode either until the next transmission opportunity or until the expected arrival of the next display update.

The algorithm has been validated for various wireless channel conditions. The impact of the ratio between the network roundtrip time and the interval between subsequent user event transmission opportunities has been investigated, as well as the influence of wireless channel path loss and the presence of background traffic between the access point and other terminals. ITRA's performance is mainly degraded by background traffic, due to collisions and packet loss complicating the prediction algorithm, whereas the impact of wireless channel path loss is only significant for a high path loss value of 90 dB. Experiments indicate that WNIC energy reductions up to 36.4 % can be achieved with actual user traces.

Acknowledgment

Part of the research leading to these results was done for the MobiThin project and has received funding from the European Community's Seventh Framework (FP7/2007-2013).

References

[1] M. Al-Turkistany, A. S. Helal, and M. Schmalz, "Adaptive wireless thin-client model for mobile computing," *Wireless Communications & Mobile Computing*, vol. 9, no. 1, pp. 47–59, Jan 2009.

[2] H. Yan, S. A. Watterson, D. K. Lowenthal, K. Li, R. Krishnan, and L. L. Peterson, "Client-centered, energy-efficient wireless communication on IEEE 802.11b networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, pp. 1575–1590, Nov 2006.

[3] S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "DYNAMO: A cross-layer framework for end-to-end QoS and energy optimization in mobile handheld devices," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 722–737, May 2007.

[4] M. Anand, E. Nightingale, and J. Flinn, "Self-tuning wireless network power management," *Wireless Networks*, vol. 11, no. 4, pp. 451–469, Jul 2005.

[5] Asus EEE, "Hardware power consumption of the Asus EEE pc," [Online] Available: http://wiki.eeeuser.com/hardware_power_consumption.

[6] P. Simoens, P. Praet, B. Vankeirsbilck, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, and P. Demeester, "Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices," in *ATNAC: 2008 Australasian Telecommunication Networks and Applications Conference*, 2008, pp. 391–396, Adelaide, Australia, Dec 07-10, 2008.

[7] R. Mangharam, R. Rajkumar, S. Pollin, F. Catthoor, B. Bougard, L. Van der Perre, and I. Moeman, "Optimal fixed and scalable energy management for wireless networks," in *IEEE Infocom 2005: The Conference on Computer Communications, Vols 1-4, Proceedings*, ser. IEEE Infocom Series, 2005, pp. 114–125, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, Mar 13-17, 2005.

[8] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: an event driven energy saving strategy for battery operated devices," in *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, Atlanta, Georgia, USA, 2002, pp. 160–171.

[9] T. Kim, J. Lee, H. Cha, and R. Ha, "An energy-aware transmission mechanism for WiFi-based mobile devices handling upload TCP traffic," *International Journal of Communication Systems*, vol. 22, no. 5, pp. 625–640, May 2009.

[10] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 power-saving mode for mobile computing in Wi-Fi hotspots: Limitations, enhancements and open issues," *Wireless Networks*, vol. 14, no. 6, pp. 745–768, Dec 2008.

[11] G. Miao *et al.*, "Cross-layer optimization for energy-efficient wireless communications: a survey," *Wireless Communications & Mobile Computing*, vol. 9, no. 4, Sp. Iss. SI, 2009.

[12] J.-L. Hsu and I. Rubin, "Cross-layer design of joint routing and rate control in ad hoc wireless networks," *Wireless Communications & MobileComputing*, vol. 10, no. 1, Sp. Iss. SI, pp. 129–144, Jan 2010.

[13] G. A. Al-Mashaqbeh, J. N. Al-Karaki, and S. M. Bataineh, "CLEAR: A Cross-Layer Enhanced and Adaptive Routing Framework for Wireless Mesh Networks," *Wireless Personal Communications*, vol. 51, no. 3, pp. 449–482, Nov 2009.

[14] C. Blanch *et al.*, "Cross-layer optimization for the scalable video codec over WLAN," in *17th International Packet Video Workshop*. IEEE, 2009.

[15] S.-L. Tsao *et al.*, "An energy-efficient transmission mechanism for VoIP over IEEE 802.11 WLAN," *Wireless Communications & Mobile Computing*, vol. 9, no. 12, 2009.

[16] I. Crk, M. Bi, and C. Gniady, "Interaction-Aware Energy Management for Wireless Network Cards," in *Sigmetrics'08: Proceedings of the 2008 International Conference on Measurement & Modeling of Computer Systems*, ser. ACM Sigmetrics Performance Evaluation Review, Special Issue, vol. 36, no. 1, 2008, pp. 371–382, International Conference on Measurement and Modeling of Computer Systems, Annapolis, MD, Jun 02-06, 2008.

[17] A. M. Lai and J. Nieh, "On the performance of wide-area thin-client computing," *ACM Trans. Comput. Syst.*, vol. 24, no. 2, pp. 175–209, 2006.

[18] F. Fok, B. Lécroart, E. Chan, P. Simoens, and B. Dhoedt, "An adaptive approach to optimize thin client protocols," in *Future Network & MobileSummit 2010 Conference Proceedings*, 2010.
[19] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper, "Virtual network computing," *Internet Computing, IEEE*, vol. 2, no. 1, pp. 33 –38, Jan/Feb 1998.
[20] T. Richardson, "The rfb protocol," RealVNC Ltd., Tech. Rep., 2009, [Online] Available:http://www.realvnc.com/docs/rfbproto.pdf.
[21] RealVNC Ltd., "Remote control software," [Online] Available: http://www.realvnc.com.
[22] K. Kaplinsky, "VNC tight encoder-data compression for VNC," in *Proceedings of the 7th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists. Modern Techniques and Technology. MTT'2001*, 2001 2001, pp. 155–7.
[23] L. Deboosere, J. De Wachter, P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Thin client computing solutions in low- and high-motion scenarios," in *Third International Conference on Networking and Services. ICNS 2007*, Jun 2007, pp. 230–5.
[24] T. Petrovic and K. Fertalj, "Demystifying desktop virtualization," in *ACS'09: Proceedings of the 9th WSEAS international conference on applied computer science*, ser. Recent Advances in Computer Engineering, 2009, pp. 241–246, 9th WSEAS International Conference on Applied Computer Science (ACS'09), Genoa, Italy, Oct 17-19, 2009.
[25] W. Qadeer, T. Rosing, J. Ankcorn, V. Krishnan, and G. De Micheli, "Heterogeneous wireless network management," in *Power - Aware Computer Systems*, ser. Lecture Notes in Computer Science, vol. 3164, 2005, pp. 137–184.
[26] "IEEE standard for Wireless LAN - Medium Access Control and Physical Layer Specification, IEEE standard 802.11, June 1999," Tech. Rep.
[27] F. Zheng, B. Gleeson, and J. Nelson, "Performance analysis and design: Power saving backoff algorithm for IEEE 802.11 DCF," in *Networking 2006: Networking Technologies, Services, and Protocols: Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, ser. Lecture Notes in Computer Science, vol. 3976, 2006, pp. 1150–1156, 5th International IFIP-TC6 Networking Conference, Coimbra, Portugal, May 15-19, 2006.
[28] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless Web access with bounded slowdown," *Wireless Networks*, vol. 11, no. 1-2, Sp. Iss. SI, pp. 135–148, Jan-Mar 2005, 8th ACM International Conference on Mobile Computing and Networking (MobiCom 2002), Atlanta, GA, Sep 23-28, 2002.
[29] Jung-Ryun Lee, Sang-Wook Kwon, and Dong-Ho Cho, "Adaptive beacon listening protocol for a TCP connection in slow-start phase in WLAN," *IEEE Communications Letters*, vol. 9, no. 9, pp. 853–5, September 2005.
[30] D. Qiao and K. Shin, "Smart power-saving mode for IEEE 802.11 wireless LANs," in *Proceedings IEEE Infocom 2005*, K. Makki and E. Knightly, Eds., vol. 3, 2005 2005, pp. 1573–83.
[31] S. Nath, Z. Anderson, and S. Seshan, "Choosing beacon periods to improve response times for wireless http clients," in *Proceedings of the second international workshop on Mobility management & wireless access protocols*, ser. MobiWac '04. New York, NY, USA: ACM, 2004, pp. 43–50.
[32] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, Aug 2000.
[33] V. Derudder, B. Bougard, A. Couvreur, A. Dewilde, S. Dupont, L. Folens, L. Hollevoet, F. Naessens, D. Novo, P. Raghavan, T. Schuster, K. Stinkens, J. W. Weijers, and L. Van der Perre, "A 200Mbps+2.14nJ/b digital baseband multi processor system-on-chip for SDRs," in *2009 Symposium on VLSI Circuits*, 2009, pp. 236–237, Symposium on VLSI Circuits, Kyoto, JAPAN, Jun 16-18, 2009.
[34] A. Dejonghe, B. Bougard, S. Pollin, J. Craninckx, A. Bourdoux, L. Van der Perre, and F. Catthoor, "Green reconfigurable radio systems," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 90–101, May 2007.
[35] P. Simoens, B. Vankeirsbilck, L. Deboosere, F. A. Ali, F. De Turck, B. Dhoedt, and P. Demeester, "Upstream bandwidth optimization of thin client protocols through latency-aware adaptive user event buffering," *International Journal of Communication Systems*, vol. 24, no. 5, pp. 666–690, 2011.