

Saving Energy on WiFi With Required IPsec

Youngsang Shin, Steven Myers, and Minaxi Gupta

School of Informatics and Computing

Indiana University

Bloomington, IN 47405, USA

shiny@cs.indiana.edu, samyers@indiana.edu, minaxi@cs.indiana.edu

Abstract. The move to a pervasive computing environment, with the increasing use of laptops, netbooks, smartphones and tablets, means that we are more reliant on wireless networking and batteries for our daily computational needs. Specifically, this includes applications which have sensitive data that must be securely communicated over VPNs. However, the use of VPNs and mobile, wireless computing creates conflicting needs: VPNs traditionally assume a stable network connection, which is then secured; in contrast, wireless computing assumes a transitory network connection due to mobility or energy-saving protocols. In this work we study the ability to use traditional VPN protocols, specifically IPsec, in mobile environments while permitting for energy savings. Energy savings come from power-cycling the wireless radio when it is not in use.

More specifically, we develop a mathematical model for determining potential power savings on mobile devices when power-cycling the radio in IPsec use settings. Next, we perform performance measurements on IPsec session resumption protocols IKEv2 [1], MOBIKE [2], and IPsec Gateway Failover (IGF) [3] to provide data for our model. We apply the model to over 3000 wireless sessions, and determine the optimal power savings that could be achieved by power-cycling the radio while maintaining an IPsec connection. We show that there is a high-potential for energy savings in the best case. Finally, we develop an efficient and simple real-world online scheduling algorithm that achieves near optimal results for a majority of users.

Key words: WiFi, VPN, IPsec, IPsec gateway failover, energy saving, security

1 Introduction

Mobile devices such as laptops, netbooks, Personal Data Assistants (PDAs), tablets and smartphones typically come equipped with WiFi and/or cellular network radios. This allows them to easily connect to the Internet, motivating their pervasive use with IP-based user applications in conducting business. According to [4], more than 50 million US workers are spending more than 20 percent of the time away from their primary workspace. Yet, with mobile users connecting over untrusted networks to organizations' sensitive resources comes the need for secure communication. Virtual Private Networks (VPN) are widely adopted and used for this purpose, and Internet Protocol Security (IPsec) [5, 6] is perhaps the most commonly used VPN protocol. Most organizations

now require that mobile employees use a VPN connection for all connections to the Internet via mobile devices for security and auditing reasons.

When an IPsec connection is built over WiFi in mobile devices, the usability of secure communication is significantly affected by two important factors: battery power and mobility. Mobile devices are presumed to be operated by battery power and studies have shown that the power usage of the WiFi interface and radio is a major fraction ($\approx 18\%$) of the power used in mobile devices [7]. Further, VPNs often require intensive use of the CPU in the computation of asymmetric cryptography while building the IPsec connection, which is known to consume considerable power. Since mobility and power-cycling radios can frequently force the asymmetric cryptographic operations to be recomputed, examining power-saving opportunities in these scenarios is crucial.

1.1 Scenarios

To help visualize issues, we consider several scenarios:

Academic at a coffee shop: An academic is writing a paper at the coffee shop. She occasionally needs access to the Internet to look up references, find papers, email, etc., but does not need continuous access, nor does she want to continuously re-authenticate to the VPN server.

Roaming Tablet User: A user with a mobile tablet computer (e.g., iPad, iPod-Touch) is traveling through a city, using available free WiFi connections whenever they appear. Since open connections cannot be trusted, a VPN connection must be maintained.

Business at the Airport: While waiting between flights, a business user gets work done. She needs to occasionally update and modify files on servers, but mostly uses local files and applications. She may have to move locations several times due to eating or gate-changes. There is no available power-plug. Her corporation requires all connections be through the VPN.

In each of these cases, the users will need to continuously reestablish VPN connections whether or not they use the VPN a frequent amount of the time. However, manually turning off the WiFi radio is probably too time consuming and frustrating to be done manually; this is especially true when a VPN connection must be re-established with each cycling. Therefore, WiFi is left on, and battery life severely reduced.

1.2 Power Saving Mode (PSM)

The 802.11 standard defines two possible operating modes for wireless interfaces: *active* mode and *power saving mode (PSM)*. In *active* mode, wireless devices can exchange data while being in receive, transmit, or idle states. The PSM is an optional mode. It lowers the energy consumption of the mobile device compared to the *active* mode. If PSM is used, a mobile client's wireless interface goes into PSM when the device is idle. Upon doing so, it informs the access point (AP) so the AP can buffer incoming data. Periodically, usually $100msec$, the AP sends a special frame, called the *Beacon*, which contains a *Traffic Indication Map (TIM)* to inform the client about

buffered frames. The clients periodically wake up and receive the Beacon to get buffered frames. A more detailed description of PSM can be found in [8].

Under our usage scenarios PSM has several problems. First, the power consumption in PSM is still close to 250mW, which is around 30% of that in *active* mode [9]. It is nontrivial and unnecessary when there is no network traffic. Second, it is hard to assume that PSM can be continuously supported when users change their location. When users move and transition across APs, the PSM support is not handed over between APs. Further, the user can encounter various APs administrated by different organizations and many old APs in use today do not support the PSM standard. For these reasons, we do not consider PSM subsequently in this paper.

A number of research papers including [10], present techniques to utilize 802.11's PSM to reduce power use. Some works [11] have suggested using proxies at the AP to look at incoming connections and respond to those connections that do not need to wake the client, and can be trivially handled by the proxy. This clearly cannot be done in the case that the data is encrypted, and the proxy is not trusted. Regardless, due to the non-trivial power consumption and inefficiencies of WiFi's PSM protocol, the works of [7, 9, 12] suggest saving power by shutting-down the WiFi radio when the WiFi interface is not actively used. These works use a lower powered radio to forewarn of an incoming WiFi signal and to awaken the WiFi interface. However, in most deployments today, such alternate low-power radios are not deployed on at least one, if not both sides of the wireless connection. We consider a situation in which the radio is predictively power-cycled, without the aid of a low-powered radio to warn of incoming traffic. This may prevent the use of push-based protocols that attempt to access a client. Nonetheless, based on our large number of wireless traces we show this is still an effective strategy for a large number of users, and power savings are achievable without additional hardware.

1.3 Our Contributions

Shutting down the radio is an effective way to save energy, but it comes with important side effects. Specifically, a disconnection in higher layer protocols and/or a change of IP address. Mobility means that the device may be communicating with many different APs over time with transition periods where no connection is available. This results in exactly the same side effects. Since one's IP address is used as a means of identification at the network layer, its change may significantly affect higher layer protocols. We are specifically interested in its effect on IPsec. When a device's IP address is modified, the IPsec connection needs to be reestablished, causing delay and energy consumption. This is particularly true due to CPU-intensive asymmetric cryptographic operations.

To the best of our knowledge, no prior works have investigated the time and energy impacts of IPsec key reestablishment and session resumption on higher layer energy preservation protocols that power-cycle the wireless radio. We consider three protocols for IPsec session resumption: IKEv2 [1], MOBIKE [2] and IPsec Gateway Failover (IGF) [3].

We develop a mathematical model for calculating the power savings possible with the use of IPsec in the presence of radio cycling under clairvoyant scheduling. In order to populate the model with appropriate parameters on timing and power usage, we compute performance measurements for both clients and servers for IKEv2, MOBIKE

and IGF and measure power usage rates and costs for different hardware and protocol executions. Next, we apply this model to over 3000 wireless sessions from Indiana university’s campus. The results demonstrate that in the presence of an IPsec connection there is a strong potential for energy savings by power cycling the wireless radio. In scenarios of interest, the hybridization of MOBIKE and IGF gives the best result, implying IGF should be considered for mobility situations. We also show that a simple and efficient algorithm can be deployed that predictively power-cycles the radio based on past network usage. We show that this algorithm achieves near-optimal results for a large fraction of users, while having minor power penalties for a negligible fraction.

Finally, we note that none of the protocols were designed for saving power on mobile devices via power cycling radios, but all three can be used as such.

2 Background on IPsec & Related Protocols

The IPsec protocol allows for private and authenticated communication over an open network. IPsec, as it is typically deployed, works in roughly two phases, a computationally intensive key-exchange phase in which asymmetric cryptographic operations are performed to share secret random keys between the client and server. Other shared state between the client and the server is also amassed and the collection is called the Security Association (SA). The SA is used to establish a secure session, through the use of symmetric key cryptography, in a second phase. Importantly, the IP addresses of the client and server are embedded in the SA, so *historically* any modifications to the server’ or client’s IP address require recomputing the first, expensive, phase to acquire a new SA.

2.1 Internet Key Exchange (IKE) protocol

IKE is the name of the asymmetric cryptographic handshake most commonly used in phase one of IPsec. Technically, there are two version of IKE: IKEv1 [13] and IKEv2 [1]. IKEv1 is inefficient and is being deprecated, all references to IKE herein refer to IKEv2. Other technical details of the protocol are beyond the scope of this paper. Importantly, MOBIKE [2] and IGF [3]—the other protocols we consider— are only designed to interact with IKEv2. IKE can itself use several asymmetric primitives (e.g., RSA or Diffie-Hellman) with different security parameters, and the choice of these parameters have an effect on performance. In our measurements we use RSA keys with 2048 bits, as RSA is the more frequently deployed primitive, and a 2048-bit key length is now considered required for many security objectives. RSA is also preferable to Diffie-Hellman due to the ability to choose small exponents, decreasing computation time at the mobile client at the expense of time on the powered server.

In regards to session resumption, by simply starting a new IPsec session from scratch each time the radio is powered on, we can of course perform a primitive form of ‘session resumption’.

Dead Peer Detection(DPD) In practice most IPsec servers run the dead peer detection (DPD) [14] protocol, to determine when clients are no longer present in order to reclaim resources dedicated to the IPsec connection. The DPD protocol is needed since many IPsec clients do not actively notify of a disconnect. However, wireless clients can be perceived as dead due to power cycling the radio or because they are in transit between APs. DPD determines whether a client is alive or not by waiting for a fixed amount of time for a reply to a keep-alive query. If there is no response, it disconnects the session, deletes any associated state and recovers any associated resources.

2.2 IPsec Gateway Failover (IGF)

IPsec Gateway Failover (IGF) [3] is an extension to IKEv2 designed for the fast, near simultaneous, resumption of a large number of interrupted IPsec connections due to server failures. It was designed to allow IPsec servers that have failed to quickly reestablish the connections with many clients once the servers are back online. This is done without needing to re-execute the computationally expensive IKE protocol with each client. The IGF protocol was not designed with mobility in mind, and we believe we are the first to recognize its potential application in mobility settings. The main scheme is based on the stateless TLS session resumption protocol [15].

In IGF, the server sends a (symmetrically) encrypted and authenticated version of its IKE SA information, called a *ticket*, to the client. The ticket's cryptographic keys are not known to the client; thus, the ticket can only be decrypted or modified by the server. The client simply stores the ticket and presents it to the server when the client needs to restore a failed connection, reestablishing the SA. Importantly, all encryption and authentication of the ticket is done strictly with *efficient symmetric key cryptographic primitives*, and thus, in a given time frame, a server can reactivate many more SAs from tickets, than it could by repeating IKE protocols. Thus, in the scenarios of interest herein, even if an IPsec server decides that a connection has been severed via the DPD protocol and deletes the session's SA, an IPsec session can be re-established via IGF.

2.3 MOBIKE

MOBIKE (IKEv2 Mobility and Multihoming Protocol) [2] is a mobility and multihoming extension to IKE, allowing IP addresses in a previously established SA to change. Thus, it enables a mobile IPsec client *that has already established a SA via IKE or IGF* to keep a connection with a server alive while changing its IP address.

MOBIKE allows mobile IPsec clients who have previously established a SA through other means, such as IKE or IGF, to change IP addresses while maintaining a connection, instead of requiring the establishment of a new IPsec connection via IKEv2. However, this protocol works only when both client and server maintain an existing IPsec connection state. In cases where the connection is lost due to DPD (or some other reason), MOBIKE cannot be used to resume a client's IPsec session. In such situations, MOBIKE can default to IKEv2 or IGF.

We therefore consider three cases relating to MOBIKE i) MOBIKE by itself, where the state of every client would need to be maintained for each client more-or-less

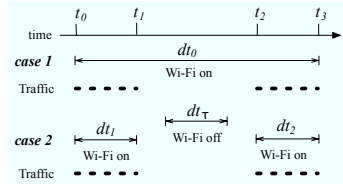


Fig. 1: Scenarios with WiFi on and off during idle times.

indefinitely (i.e., we assume DPD is not run and connections live forever); ii) MOBIKE+IKEv2 and iii) MOBIKE+IGF, where in the latter two cases it is assumed that sessions that are inactive for short periods of time (several minutes) are reinitiated with MOBIKE, but sessions that are inactive for longer periods of time, and thus likely disconnected by the server, are resumed through IKEv2 or IGF respectively.

3 Power Savings for Session-Resumption Protocols

We develop a deployable mathematical power-saving model for IPsec over WiFi for different session resumption protocols. Next, we use it to analyze real world wireless sessions to determine the potential for the power saving schemes. We show that in the optimal case substantial power savings can be achieved using clairvoyant scheduling. We realize that such omniscient optimal scheduling is not possible in real life, but if optimal scheduling does not provide savings there is no potential for a more limited algorithm. Our results provide an upper bound on potential power savings. Finally, we present a practical online scheduler that achieves near optimal power-savings.

3.1 Mathematical Model

In Figure 1 we consider two cases. In case 1 the WiFi radio is on from time t_0 to t_3 but has a large idle gap between t_1 and t_2 , and therefore could have potentially been switched off to save power. Equation 1 gives the cost of case 1 for time period $dt_0 = t_3 - t_0$, where $WiFi_M$ represents the maintenance cost per second of keeping on the WiFi radio while it is idle.

We are interested in cases in which the WiFi radio is instantly switched off if it were going to stay idle for some period longer than dt_τ , where dt_τ represents the minimal amount of time a radio needs to be off for it to have been cost effective in terms of power consumption. Note that the power savings have to account for the costs of cycling the wireless radio off and on, and then reestablish an IPsec connection.

Let $WiFi_C$, $IPsec_C$, $WiFi_D$ and $IPsec_D$ represent the respective energy costs of connecting and disconnecting WiFi and IPsec connections, where $IPsec_C$ and $IPsec_D$ are dependent on the session resumption protocol in question. Equation 2 calculates the break-even time, $dt_{\tau'}$ of switching the wireless radio on and off and resuming an IPsec session *assuming that setting up IPsec and WiFi connections are instantaneous actions*. In reality, there is a time associated with each, call them $WiFi_t$ and $IPsec_t$, and during

dt_τ	idle-time/power savings cross over point
$WiFi_M$	idle WiFi maintenance cost per second
$WiFi_C$ & $WiFi_D$	WiFi startup and shutdown costs
$IPsec_C$ & $IPsec_D$	IPsec connection and disconnection costs
$WiFi_t$ & $IPsec_t$	time to establish WiFi and IPsec connections

$$dt_0 \cdot WiFi_M \quad (1)$$

$$dt_{\tau'} = \frac{\sum_{i \in \{C,D\}} (WiFi_i + IPsec_i)}{WiFi_M} \quad (2)$$

$$dt_\tau = dt_{\tau'} + (WiFi_t + IPsec_t) \quad (3)$$

$$\sum_{i \in \{C,D\}} (WiFi_i + IPsec_i) \quad (4)$$

$$+(dt_1 + dt_2) \cdot WiFi_M$$

that time the device is wasting extra base-line power at a rate of $WiFi_M$ to remain on. So, to truly capture the costs one calculates dt_τ , as in Equation 3.

In Figure 1 case 2, we denote the same traffic scenario as case 1, but with a power saving scheduler. The idle time occurs between t_2 and t_1 , since $(t_2 - t_1) > dt_\tau$ the IPsec connection is disconnected and the wireless radio is switched off to save power. At time t_2 , the WiFi radio is enabled again as the client needs to use the WiFi connection again. When the WiFi interface is enabled, the VPN connection needs to be reconnected. We consider the reconnection costs of the IPsec connection via IKE, MOBIKE or IGF. The cost of case 2 is given by Equation 4 where $dt_1 = t_1 - t_0$ and $dt_2 = t_3 - t_2$. We do not charge for a user's network traffic during dt_1 and dt_2 since they are the same in both cases. Clearly, *whenever Equation 4 is valued strictly smaller than Equation 1, the power saving scheme is beneficial.*

3.2 Patterns in Real-World WiFi Traffic

To investigate how effective radio-cycling power saving schemes can be, we collected one day of anonymized NetFlow data from Indiana University's *wireless APs*. The data spans 3098 client connections to the *wireless APs* and contains packet headers, packet sizes, and timing information for IP packets going through them. For privacy reasons, the packet headers have source and destination IPs anonymized but port numbers are intact. In cases where only the start and stop time for a flow of multiple packets is known, we assume that the packets were distributed evenly over that time. This is a worst-case assumption that underestimates the potential times the wireless radio can be deactivated in any schedule. It should be noted that no effort was made to have any of the clients run any traffic shaping protocols to optimize the amount of time the radio could be disconnected. The data represents average wireless usage of mobile users at our University.

In Figure 2 we present two sample idle-time patterns that we collected by subjecting network traffic generated by a few real-world users using Wireshark. They show the very different patterns wireless traffic can take for different users. We note that the idle time between most packets, for almost all users, is well under a minute in each case but there are large gaps of several minutes between some of the packets of many users', as is demonstrated by User 1 in the figure. This implies significant power savings are conceivable for such users. With the longest gap between packets of User 2 being 21 seconds, power savings are less likely for users with such behavior. Our goal is to save energy in the cases of users like 1, while not penalizing users with usage patterns similar to 2.

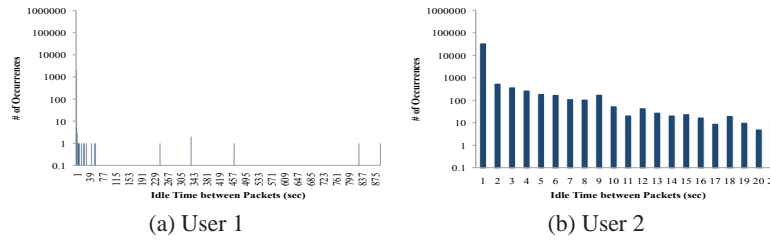


Fig. 2: Idle time between packets for two users over an hour period.

Device	CPU/RAM	Network Connection	Operating System
Client	Pentium M 1.86Ghz/512M	Intel Pro/Wireless 2200bg (802.11g)	Linux kernel 2.6.18
Server	Pentium IV 1.3Ghz/640M	10Mbps Ethernet	Linux kernel 2.6.22

Table 1: Configuration of machines used for measuring energy usage.

3.3 Measuring Variables for the Mathematical Model

To estimate power savings with our model, we need to estimate the cost of each variable in Equations 1 & 4. Works in [7, 9, 12] used multimeters to gauge power usage. Instead, we determine the battery usage by querying the mobile device through the Linux Advanced Configuration & Power Interface (ACPI) [16]. ACPI is an abstraction layer that enables OS-directed configuration, power, and thermal management of mobile, desktop, and server platforms. ACPI does not provide as high a resolution of energy measurement as multimeters, but it is simpler to measure and it might be argued is a more effective tool for evaluating our benefits of power saving scheme, because *it represents the information that a typical operating system, and thus scheduling algorithm, will have to measure power-usage*. We use standard laptops to perform energy measurements. Specifics of the configuration of devices are shown in Table 1.

The resolution of the Linux ACPI measurement may be deemed low due to the fact that it cannot directly obtain the power usage of each operation independently. This is due to overhead of the operating system, and other processes running on the machine during calls to the API. Therefore, we measure the average energy usage over 25 iterations for each of the operations. We measure what the API returns as the differential between the battery life before and after the operation. Of course, we must normalize these values to compensate for the fact that there is a base-line power consumption for each mobile device. Through repeated measurements, we found that the baseline rate of energy consumption for the laptop was approximately 18Watts when it was on with the WiFi interface off, and when its LCD's brightness was set to the highest value. This baseline power-consumption is needed, since the proposed power saving scheme incurs an additional waiting time as the WiFi and IPsec connections are established and torn-down and one must charge the power-saving schedule for the extra baseline power consumed during these times. For $WiFi_M$, we measure the energy usages with and without the WiFi connection for 10 minutes. We conduct this experiment 25 times and then calculate an average.

Operations & Steady States	Cost
$WiFi_M$	2.29 W
$IPsec_C$ & $IPsec_D$ with IKE	13.14 J
$IPsec_C$ & $IPsec_D$ with IGF	9.2 J
$IPsec_C$ & $IPsec_D$ with MOBIKE	0.40 J
$WiFi_C$ & $WiFi_D$	46 J
$WiFi_t$	2 seconds

Fig. 3: Energy usages obtained through Linux ACPI and time for WiFi association with AP.

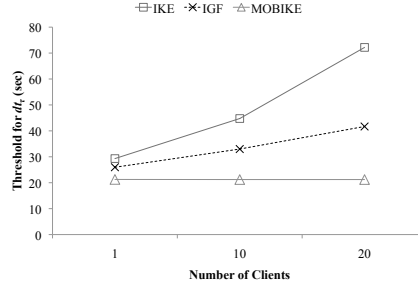


Fig. 4: Changes in threshold dt_τ for different session resumption protocols as the number of concurrent connections increase.

We measured the costs of $WiFi_C$ & $WiFi_D$ and $WiFi_t$ at six different locations and APs to determine if the costs depend on the configuration of APs. The locations include libraries, restaurants, coffee houses, and book stores. The results are shown in Figure 3. For 5 of the 6 locations the costs of $WiFi_C$ & $WiFi_D$ were within $46 \pm 1J$ and $WiFi_t$ was within $2 \pm 0.5 seconds$. An outlying data point, at $61J$ for $WiFi_C$ & $WiFi_D$ and at $4.5 seconds$ for $WiFi_t$, seemed to be due to the fact that there were many APs with the same Service Set Identifier (SSID) at the location. Elsewhere, there were many SSIDs, but each was served by only one AP. We assume that $46J$ for $WiFi_C$ & $WiFi_D$ and $2 seconds$ for $WiFi_t$. Unlike the measurements for $WiFi_C$ & $WiFi_D$ and $WiFi_t$, we measured the cost of $IPsec_C$ & $IPsec_D$ at only one location because these costs are independent of the WiFi client and its relation to the AP. These measurements were averaged over 25 runs. Lastly for $IPsec_t$, we use the latency numbers in Figure 10. Figure 3 presents the measured values of model variables.

We calculated the threshold value dt_τ for IKE, IGF, and MOBIKE as defined by Equation 3 using the measured values for $IPsec_C$, $IPsec_D$, $WiFi_C$, $WiFi_D$, $WiFi_M$, $WiFi_t$, and $IPsec_t$. The only location dependent values are $WiFi_C$, $WiFi_D$, and $WiFi_t$, and since they do not appear to vary much, dt_τ is effectively location invariant. The value $IPsec_t$ is dependent on the load on the IPsec server: the higher the server load, the longer the latency experienced by the client (c.f. Sec. 4). While the client is latent and waiting for a server response to the IPsec protocol, it pays a cost of energy at the rate of $WiFi_M$. Figure 4 summarizes the approximate thresholds for each protocol for 1, 10 and 20 simultaneous clients. As expected, the threshold increases with the number of clients for all protocols but MOBIKE, which has no real computational costs.

3.4 Power Savings Under Optimal Scheduling

We investigated the optimal power savings for various session-resumption protocols using the 3098 WiFi traffic logs. We assumed that the scheduler deactivates the wireless card immediately upon detecting inactivity and invokes it so it is functioning just in time for a packet to arrive, assuming such power-cycling saves power.

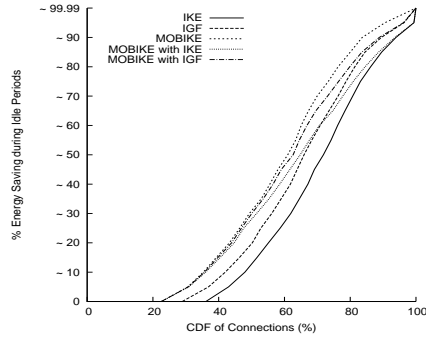


Fig. 5: Energy savings for real-world WiFi sessions under optimal scheduling with different session-resumption protocols.

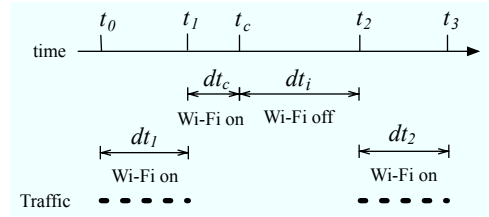


Fig. 6: Idle-time scenario for the prediction algorithm.

Recall that in our model, power savings depend on dt_τ , which in turn depends on both the number of concurrent connections the server is handling, and the AP that the user is accessing. We consider three cases where the IPsec server is serving 1, 10, and 20 concurrent connections. We chose 20 as the maximal number of concurrent clients due to the limitation of the strongSwan software¹, which does not permit larger configurations than 20. We consider five variants of session-resumption protocols: i) IKE, ii) MOBIKE, iii) IGF, iv) MOBIKE+IKE and v) MOBIKE+IGF. With the hybrid protocols iv) and v) we assume that MOBIKE is used to resume sessions that are idle for less than 2 minutes and then the alternate protocol (IKE or IGF respectively) is used otherwise. We chose 2 minutes as it is the default disconnect time for the DPD protocol in strongSwan. To compute model variables for hybrid cases, we use the appropriate $IPsec_C$, $IPsec_D$, and $IPsec_t$ dependent on the protocol for session resumption.

In the case where the VPN server is handling 20 concurrent session-resumption requests, we note that 78% of WiFi connections reap some benefits of power savings. The differences among the various session-resumption protocols is shown in Figure 5. The cumulative distribution function (CDF) of the percentage of connections that reap energy savings is given. As expected, MOBIKE has the best performance but this is under the unrealistic assumption that the DPD protocol never terminates sessions. DPD is in fact used to release sessions specifically so servers do not become overwhelmed reserving resources for dead connections. However, when MOBIKE is combined with IGF (and DPD), we get only slightly worse performance than MOBIKE by itself. We skip presenting the results for 1 and 10 concurrent clients for brevity but note that as the number of concurrent clients drops to one, the CDFs for all of the protocols cluster close to the MOBIKE curve.

¹ *strongSwan* is an open source IPsec implementation for Linux.

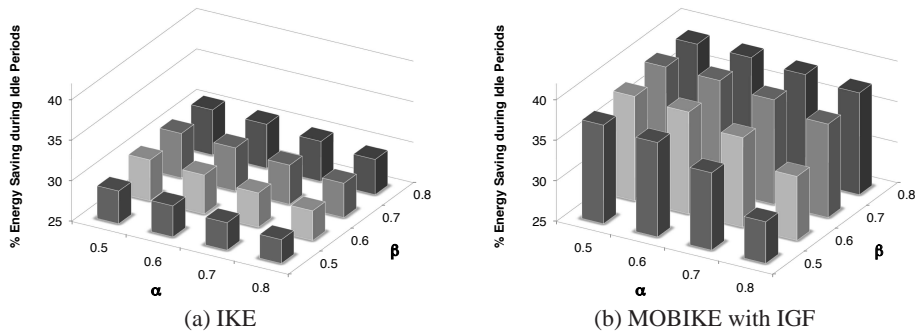


Fig. 7: Average energy savings for 100 WiFi sessions using worst and best session-resumption protocols under varying α and β .

3.5 A Real-World Scheduler

Having shown that optimal scheduling permits potentially great energy savings, we now provide a simple prediction algorithm that uses a user’s previous network usage history to predict future network usage requirements, and to decide if and when the radio should be power-cycled. When the radio is on but idle we estimate the probability that the tentative remaining idle time is longer than dt_τ , the minimal time period the radio can be off to save power. If the estimated probability is higher than a given pre-defined threshold α , the radio is power-cycled. In cases where there is no historical data yet, say during device initiation, the protocol uses a threshold value determined by calculating the average of users’ history on our real world data traces. We re-enable the WiFi radio only when the client again attempts to send a packet. This can result in lost incoming packets if the WiFi radio is off. We consider two scenarios: i) all such packets are lost, and ii) the IPsec server acts as a proxy for basic networking protocols.

To help demonstrate the scheduler operation, we depict a scenario in Figure 6. Here, an idle time starts at time t_1 . The current time is t_c , and we must make a decision as to whether or not to power-cycle the radio. The prediction algorithm estimates the probability, P_{t_c} that the tentative remaining idle time dt_i will be longer than dt_τ , based on historical network usage patterns. If the estimated probability is greater than a threshold value α the WiFi is turned off.

The probability, P_{t_c} can be calculated in Equation 5. When it is greater than a threshold, α in Equation 7, the WiFi radio is turned off. One issue with the proposed scheme is that if a user’s network usage history does not contain any idle period whose length is longer than at least dt_τ , P_{t_c} is always 0. For this case, we adopt a probability, P_{avg,t_c} calculated from average users’ network usage history in Equation 6. Thus, when P_{t_c} is 0, the algorithm checks if P_{avg,t_c} is greater than a threshold, β in Equation 8. The values are tabulated by taking into account all of the network trace-data we have, minus 100 traces we have randomly chosen and separated to use for independent performance evaluation.

Scheduler Performance To evaluate the scheduler, we chose 100 different sessions in our wireless traces uniformly at random. We also calculated the average users’ network-

t_c	current time
m_{t_c}	# of idle periods in user history where length $\geq dt_c$.
n_{t_c}	# of idle periods in user history where length minus $dt_c \geq dt_\tau$.
$m_{avg.t_c}$	# of idle periods in average user's history where length $\geq dt_c$.
$n_{avg.t_c}$	# of idle periods where length minus $dt_c \geq dt_\tau$.

$$P_{t_c} = n_{t_c} / m_{t_c} \tag{5}$$

$$P_{avg.t_c} = n_{avg.t_c} / m_{avg.t_c} \tag{6}$$

$$P_{t_c} > \alpha \tag{7}$$

$$P_{avg.t_c} > \beta \tag{8}$$

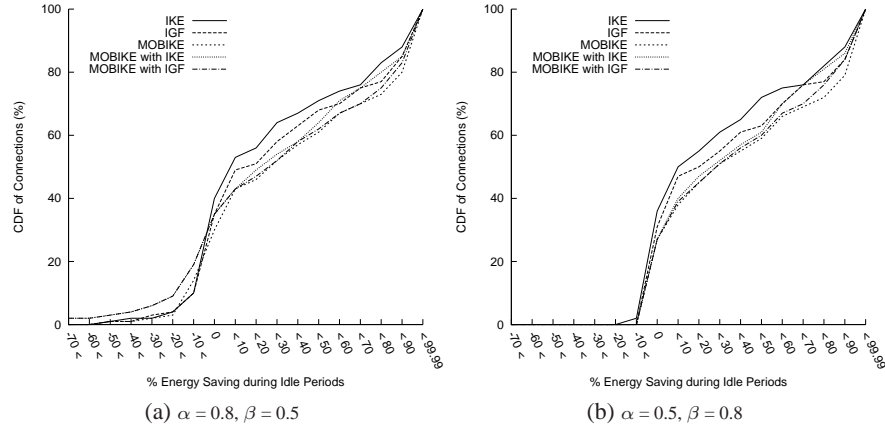


Fig. 8: CDF of percentage energy savings under the history-based scheduling with different resumption protocols

usage history from all the sessions excluding the 100 test sessions. We simulate α and β with values 0.5, 0.6, 0.7, and 0.8, where the greater value the more conservative our power-saving is. In Figure 7, we present the average energy savings for worst (IKE) and second-best (IGF+MOBIKE) of the resumption protocol we considered. MOBIKE has the best savings, but the need to keep an unlimited number of sessions open is unrealistic and thus discounted as a potential protocol. As α increases, the average savings decrease. This means that the conservative prediction misses energy-saving opportunities based on user’s historical usage patterns. In contrast, as β increases (the threshold based on average user’s data), better average savings are achieved. This is because an average users’ network-usage history does not reflect each user’s usage accurately and the conservative prediction avoids false positives.

In Figure 8, we show the percentage energy savings with different resumption protocols when α and β are 0.8 and 0.5, and 0.5 and 0.8 respectively. In Figure 8a, When β is 0.5, less than 15% of users actually spend more energy due to such false positives in the prediction. However, when β is 0.8 in Figure 8b, only around 2% of users spend less than 10% more energy under IKEv2 protocol. We get the best energy saving results with $\alpha = 0.5$ and $\beta = 0.8$. *The prediction algorithm achieves energy savings that are within 3.5% difference from the optimally-scheduled case. That is, it achieves over 90% of the maximum energy savings which can be obtained in the optimal scheduling.*

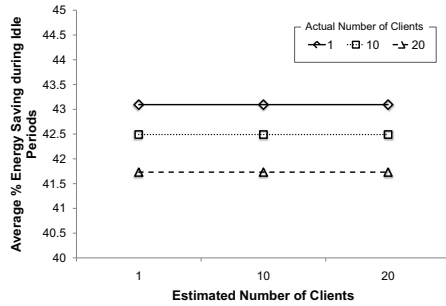


Fig. 9: Average energy savings on real-world WiFi session for potential mis-estimation of the number of concurrent clients in MOBIKE+IGF.

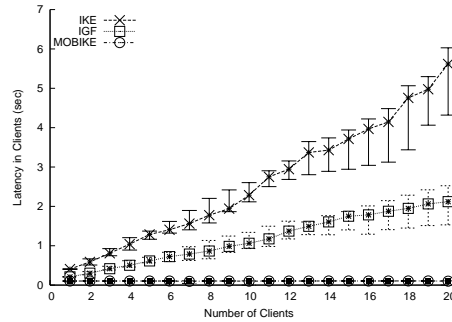


Fig. 10: Client latencies for various session resumption protocols.

Estimating dt_τ : The scheduler needs access to several measured values. The measured energy usage values in Figure 3 are relatively constant and can be embedded in to the client. On the other hand dt_τ depends on the IPsec server’s load. To estimate an effective dt_τ , the client should be able to approximate the server’s current load. However, it makes the client implementation too complicated for the client to measure and maintain such server’s response time. Instead, the IPsec server can efficiently measure statistics on the number of concurrent clients it serves over time and broadcast this information to the clients. For example, the server can generate semi-hourly statistics estimating its loads based on previous usages on previous days and send them to the client when the client establishes an IPsec connection to the server for the first time each day. However, the estimates might be off, resulting in over- or under-estimates due to sporadic or unusual activities at the server. Thus, it is crucial to investigate what effect a false estimation of dt_τ would have on the energy savings or costs for the client. Figure 9 shows the effects of such false estimation. Due to space restrictions, we show only IGF+MOBIKE, which can be seen to be fairly immune to poor estimation of dt_τ . This is because an approximate threshold value for dt_τ changes very slowly in IGF as the number of concurrent clients is changed (Figure 4). Further, in MOBIKE, it is almost invariable.

Efficiently Computing Variables for the Scheduler: In practice a network card could not keep track of a data-structure that maintained all historical idle-times longer than dt_τ , and then compute P_{t_c} on the fly. However, in practice there are only two issues one cares about i) the value of t_c for which $P_{t_c} = \alpha$, as a network card can just wait until it is idle for such a time t_c , and then power-off the radio; and ii) how to determine the new value t_c such that $P_{t_c} = \alpha$ when a new idle period longer than dt_τ is recorded. This can be done by discretizing time, and making a set of buckets, one for each discrete time interval. We have a counter for each bucket to represent the number of idle-times whose length fell in to the interval defined by the bucket. When a new idle time of length t is processed, we increment the bucket corresponding to t .

We must simultaneously keep track of which bucket has the $\alpha \cdot N$ th element (and thus which bucket corresponds to the time interval t_c for which $P_{t_c} = \alpha$). For purposes of example let $\alpha = 0.5$. We need to keep track of the bucket containing the median. This is done by keeping an index to the current bucket containing the median, the size of that bucket, and the position within that bucket of the median. If a new idle length is put in a smaller (larger) bucket than the median bucket, the relative position in the bucket is increased (decreased resp.) by one. If the new idle length goes in the same bucket as the median, then its index is increased by $1/2$ and the bucket size is increased by 1. Should the index go below 0, or above the bucket-size, the median is moved to the next bucket and the process repeated.

False Negatives & Positives: It is important to calculate the false positives (missed opportunities to cycle the radio) and false negatives (the radio is power-cycled, but incoming packets are dropped) of our scheduler. Due to space limitations, we provide these for the MOBIKE+IGF scheduler only. The false positive rate is quite low and depicted in missed power-cyclings per session in Figure 12, with 90% of the sessions have fewer than 10 false positives. This leaves little room for improvement.

Since the radio is only power-cycled back on when the client sends packets, incoming packets can be lost. They are the false negatives. We consider two scenarios: i) all missed packets are simply dropped, and ii) the IPsec server acts as a simple proxy responding for the client to simple network protocols that do not actually need the client's presence. Specifically, it responds to DHCP, DNS, and MICROSOFT-DS. Such proxies have been well studied when used at AP's [11], but their use must be incorporated in to the IPsec server due to the encrypted traffic. In Figure 11 we show the results. We clearly see that 60% of users suffer no dropped packets in scenario i), and 70% of users suffer no dropped packets in scenario ii) with the proxy. Without specific packet information in the traces, it is impossible to determine how important the missed packets were in the 30% to 40% of cases we missed at least one packet. However, it suggests that a simple user-interface can be devised that allow users to quickly switch from an energy-saving mode of IPsec to a performance mode, where users can quickly decide if they want power-savings. In such scenarios, users can default to power-savings, and if they notice performance issues they can toggle the device to an alternate mode. We consider more advanced schedulers as future work.

4 Comparison of Performance of IPsec Session-Resumption Protocols

We evaluate and compare the performance of the IGF, IKE and MOBIKE protocols, from both the server and client perspectives, as the number of concurrent connections that are actively trying to be established (or re-established) increases. This comparison not only provides *the first performance data on concurrent use of the IGF protocol*, it also provides essential validated data on which to base the mathematical power saving model in Section 3.

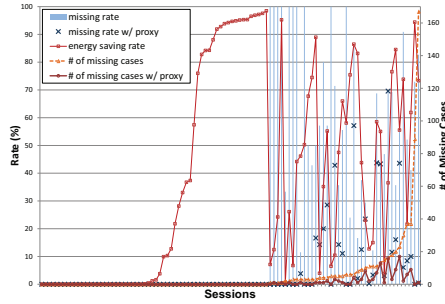


Fig. 11: Comparative depiction of energy savings and missed packets in the 100 test sessions.

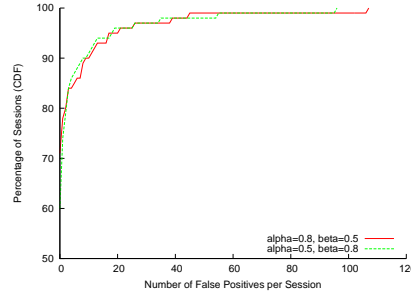


Fig. 12: CDF for false positives per session.

Device	CPU/RAM	NIC	Operating System
Server	2.8Ghz, 2G	100Mbps	Linux kernel 2.6.25
Client 1-20	3.2Ghz, 2G	100Mbps	Linux kernel 2.6.18

Table 2: Configuration of machines for multiple client connection measurements.

4.1 Methodology

We implemented the IGF protocol as an extension to strongSwan [17], which is an open source IPsec implementation for Linux. StrongSwan already implements traditional IKE and its extension MOBIKE. For our evaluation, we use two different versions of strongSwan: 4.2.4 for IPsec clients and 4.1.8 for an IPsec server²

For the experiments, we used twenty one x86 Dell Optiplex GX Pentium IV machines which were connected through Ethernet switches. Table 2 shows their specifications. To observe the effects of connection (re)initiation on an overloaded server, we chose an inferior hardware configuration for our server so it was easier to stress. Although the measured latencies include message transfer times over the network, such factors are not important for comparisons between different server loads as the communication costs stay essentially constant. This is because the network is far from capacity, while the computational load of the server increases. We perform experiments over the wired network to obtain (nearly) constant latencies. For IKE security settings, we used a 2048-bit RSA key and a 128-bit AES key, a minimum requirement in today’s security contexts. We considered cases of multiple clients simultaneously (re)connecting to the IPsec server. We report average latency over 25 independent experiments.

² We tested various versions of strongSwan between 4.1.8 and 4.2.5. However, only the given combination allowed up to 20 *simultaneously initiating* clients.

4.2 Performance Evaluation

Figure 10 gives average *client* latencies for up to 20 concurrent connections in IKE, IGF, and MOBIKE respectively. In the case of IKE and IGF, the data points represent latencies incurred when concurrent connections are being formed or reformed respectively. In the case of MOBIKE, we see latencies involved in updating the IP address associated with a client without establishing a new session. MOBIKE incurs the least latency. This is expected since MOBIKE needs to only update the client IP address and does not perform any cryptographic operations, symmetric or asymmetric. Its very low overhead enables the latency to stay essentially constant irrespective of the number of concurrent clients. As a result of avoiding computationally expensive asymmetric key cryptographic operations, the latency of IGF is also almost half that of IKE when only a single client is connecting to the server. But the gap between IKE and IGF increases with the number of concurrent clients. This data shows that re-connection through IGF easily outperforms IKE, with increasing performance benefits as the number of simultaneously initiating clients increases.

The *server side* processing times of the IPsec server for concurrent IPsec client connections are not shown due to space constraints. However, as expected, they are nearly identical to client latencies, but with tighter variance due to the lack of noise in the network measurements. The processing times were measured by computing the difference between time-stamps of the first network message to arrive from a client to the end of last operation of the protocol for said client.

5 Related Work

A formal security analysis of IGF and a evaluation of the performance of a prototype implementation is presented in [18]. The authors simulate a large number of connections with file transfer, as opposed to actual concurrent connections. Hence, the evaluation does not effectively demonstrate the performance of IGF and IKE with concurrent clients.

For fast IPsec reconnections, work in [19] introduced zero address-set functionality (ZASF) for MOBIKE. ZASF allows a mobile client with a predicted long idle period to notify the IPsec server that it will temporarily disable its radio. The server acknowledges, and temporarily disables all related IPsec states, concurrently dropping any packets destined to the client. This approach is similar to our approach (excluding our proxy), but further requires indefinitely Security Association storage for each of its clients and requires the disabling of DPD and the enabling of a local policy which terminates connections that are disabled for too long of a period. The IPsec server must also store all associated SA state information for all clients that are asleep. Our scheme, in comparison would use IGF to recover from sessions terminated by DPD, and no SA state information need be stored at the server when clients are disabled.

[20] proposes a scheme to transfer an IPsec connection state through Context Transfer Protocol (CXTP) [21] when a mobile IPsec client needs to connect to a different gateway due to changes in physical and thus network location. This scheme does not consider the case where the initial gateway does not maintain an IPsec connection due

to transient connections on mobile clients. It therefore does not deal with mobile clients as we consider herein.³

[22] extends Multi-Layered IPsec (ML IPsec) [23] to support mobility by integrating ML IPsec with Mobile IP as presenting an efficient key distribution protocol between Foreign Agents (FA) for Mobile IP. However, they do not discuss the issues in building trust between FAs managed by different organizations; furthermore, they do not support a fast IPsec reconnection after relatively long absence in network.

A number of research papers, such as [10], present a technique to effectively utilize power saving mode (PSM) in the 802.11 WiFi standard [8] to reduce radio energy usage by clients. The work of [10] presents a scheme in which PSM can effectively be used to save time and energy costs by switching power modes. However, this work assumes that each network application will provide somewhat accurate predictions about its network usage. Based on the predictions the scheme decides when to activate the PSM. However, due to non-trivial power consumption and inefficiencies of the PSM itself, [7, 9, 12] propose different power saving strategies that turn off the WiFi interface when it is not used, and concurrently activate a lower power/lower bandwidth radio. This radio is used to signal the activation of the WiFi radio when there is traffic to be communicated. The work of [9] utilizes Bluetooth as the low power radio not only for activating a WiFi connection, but also for low bandwidth data transfer where WiFi is not needed. However, the power saving schemes in [7, 9, 12] require a dual access point as an infrastructure, making incremental deployment difficult. Furthermore, none of them have considered the effect on the connection-oriented security protocols such as IPsec.

A number of research papers [11] have considered using proxies at APs to allow APs to drop or respond to traffic that is destined for sleeping clients. In the IPsec scenario, such proxies must be at the IPsec server, since the AP would be unable to read the encrypted packets.

6 Conclusions

Our results show the clear and practical potential for energy savings on mobile devices through power cycling the wireless radio, even in the presence of mandatory IPsec connections. While IPsec allows some energy savings, servers that handle many clients do much better when they consider more appropriate protocols. While using MOBIKE by itself, without dead-peer detection (DPD), gives the best potential savings, this scenario seems rather unlikely to be practical in large organizations due to the potentially large state and committed resources a server would need to maintain with many clients. For large organizations, MOBIKE+IGF hybrid represents a close second in terms of performance, even when a server needs to continuously handle multiple concurrent session resumptions. In fact, a simple implementation of our scheduling algorithm achieves extremely rare energy-penalties, but over 50% of network sessions could save 20% or

³ In cases where both servers are for the same organization and implicitly trust each-other, then by having a common secret keys between servers, IGF could resume sessions between gateways.

more of their wireless energy costs for idle times! Further, the algorithm performs nearly optimally in terms of potential energy savings. Simple modifications to the servers to broadcast estimates on their load, and clients to predict when they power-cycle their radios are relatively easy to implement in either hardware or software.

References

1. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. RFC4306 (December 2005)
2. Eronen, P.: IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC4555
3. Sheffer, Y., Tschofenig, H., Dondeti, L., Narayanan, V.: IPsec Gateway Failover Protocol. draft-sheffer-ipsec-failover-04.txt (July 2008)
4. Palumbo, S., Dyer, N.: Maximizing Mobile Worker Productivity. Yankee Group Research, Inc (January 2008)
5. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC4301 (2005)
6. Housley, R.: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC4309 (December 2005)
7. Agarwal, Y., Schurgers, C., Gupta, R.: Dynamic power management using on demand paging for networked embedded systems. In: ASP-DAC. (2005)
8. IEEE Computer Society: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE Standard 802.11, 1999 Edition (1999)
9. Pering, T., Agarwal, Y., Gupta, R., Want, R.: CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces. In: ACM MobiSys. (2006)
10. Anad, M., Nightingale, E.B., Flinn, J.: Self-Tuning Wireless Network Power Management. In: MobiCom. (2003)
11. Nedevschi, S., Chandrasheka, J., Liu, J., Nordman, B.: Skilled in the art of being idle: Reducing energy waste in networked systems. In: ACM/USENIX NSDI. (2009)
12. Shih, E., Bahl, P., Sinclair, M.J.: Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In: ACM MobiCom. (2002)
13. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). RFC2409 (November 1998)
14. Huang, G., Beaulieu, S., Rochefort, D.: A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers. RFC3706 (February 2004)
15. Salowey, J., Zhou, H., Eronen, P., Tschofenig, H.: Transportation Layer Security (TLS) Session Resumption without Server-Side State. RFC4507 (May 2006)
16. Linux/ACPI project: Linux ACPI. <http://www.lesswatts.org/projects/acpi>
17. strongSwan project: strongSwan. <http://www.strongswan.org/>
18. Tegeler, F.: Security analysis, prototype implementation, and performance evaluation of a new IPSec session resumption method. Master's thesis, University of Goettingen (2008)
19. Kivinen, T., Tschofenig, H.: Design of the IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC4621
20. Allard, F., Bonnin, J.M.: An application of the context transfer protocol: IPsec in a IPv6 mobility environment. IJCNDS **1**(1) (2008)
21. Loughney, J., Nakhjiri, M., Perkins, C., Koodli, R.: Context Transfer Protocol (CXTP). RFC4067 (July 2005)
22. Choi, H., Song, H., Cao, G., Porta, T.L.: Mobile multi-layered IPsec. In: IEEE Infocom. (March 2005)
23. Zhang, Y., Singh, B.: A multi-layer IPsec protocol. In: USENIX Security Symposium. (August 2000)