Crosslayer Firewall Interaction as a Means to Provide Effective and Efficient Protection at Mobile Devices

Peter Langendoerfer[†], Krzysztof Piotrowski[†], Steffen Peter[†], Martin Lehmann[‡]

[†]IHP, Im Technologiepark 25, 15236 Frankfurt (Oder) , Germany
langendoerfer@ihp-microelectromics.com
[‡]DFS Deutsche Flugsicherung GmbH, Langen, SH/IR, Am DFS-Campus 2, 63225 Langen, Germany

Abstract. In this paper we discuss packet filtering firewalls and an application level gateway approach used to secure handheld devices. We propose a firewall management plane as a means for cross layer interaction. In our approach the application level gateway updates the firewall rules based on its knowledge about whether or not a certain source is sending malicious packets. Hereby we pursue a policy of removing malicious packets as close as possible to the network interface. We show that in case of secure web service such a cross layer interaction can significantly decrease the CPU load in case of attacks, i.e., if many malicious packets arrive at the handheld device. Our measurement results show that our cross layer approach can reduce the CPU load caused by the application layer gateway by about 10 to 30 per cent. Finally we propose an integrated firewall processing approach that promises further improvements. It integrates the application controlled firewall before the MAC and provides crosslayer mechanisms to reduce the performance issues of traditional firewall approaches.

1. Introduction

Motivation Mobile devices are becoming more and more powerful, e.g. current HP iPAQs are equipped with 400 MHz Xscale processors and 128Mbyte memory. In addition wireless modems are integrated into these devices. With theses increasing capabilities it becomes feasible to integrate mobile devices into e-commerce architectures, e.g. in business-to-consumer and in business-to-employee applications, too. As a result of this development the amount of sensitive data that is stored on mobile devices will increase tremendously. So, mobile devices will attract an increasing number of attackers, and since their integration will be done in an 'all-IP' approach, they are exposed to all typical Internet attacks. From our point of view mobile devices are tempting target due to the following facts:

1. The medium (air) can easily be accessed by anyone.
2. Mobile devices are not protected by additional hardware or software such as firewalls, which are normally deployed at the border of a company network.
3. Compared to a fully-equipped PC or laptop, the mobile devices such as PDAs and mobile smart-phones still have very limited resources (calculation and battery power). Strong and exhaustive use of security means drains down the battery of the mobile device quite fast leading to inconvenient up times.

Thus, mobile devices are normally not as protected as more powerful devices are. The first two points cause the device to be immediately exposed to the attacker and the third one limits the ways to defend it. Our approach uses layer interaction in order to reduce the computational burden, caused by security mechanisms. This also leads to the fact that the mobile device is protected against malicious communication partners at the lowest possible layer, which in turn increases the security level of the mobile device.

Contribution and structure of this paper In this paper we discuss the performance of firewall approaches running at MAC, IP and application layer on a mobile device, i.e. a state of the art PDA. The MAC and the IP layer personal firewalls do simple packet filtering, whereas the application level gateway that does content inspection in order to secure Web Services. Our measurements clearly indicate that simple packet filtering firewalls can be used without a significant degradation of the performance and up time of the mobile device. The Web Service Security approach [8] results in a much higher computational burden for the mobile device, which is due to its intensive use of cryptographic means.

In order to decrease the effort spent per malicious packet we realized a cross layer interaction between the IP netfilter and the Web Service security gateway. The latter updates the IP packet filter tables with new IP addresses as soon as it identifies a certain source as malicious. Our measurements show that this approach reduces the computational burden of the mobile device by 75 per cent in case of application-layer attacks. We also investigated the potential benefit of extending our cross layer approach down to the MAC layer. Here our performance comparison of IP netfilter and EBtables clearly indicated that the major effort is not due to firewall processing but to base band and physical layer processing. In order to reduce computational load also on these

lower layers we propose a protocol architecture that integrates MAC and IP functionality with filtering of malicious packets, and that supports interaction with the base band processing.

This paper is structured as follows. Section 2 provides a short state of the art. In the following section we discuss the idea of layer interaction that provides efficient but lightweight security architecture for mobile devices. In section 4 we present our measurement results. Then we present the idea to filter the packets before the MAC layer and its evaluation. The paper concludes with a short summary and an outlook on further research.

2. Related work

Cross layer interaction is an important research issue in the recent year. The basic idea of this approach is to reduce the computational load of mobile devices by exchanging layer specific information, so that more sophisticated decisions can be taken at a higher or lower layer [18, 19, 20, 21]. These approaches are concentrating on the protocol layers two to four, i.e. they are agnostic with respect to the application running. There are some specialized approaches, which gain additional benefit from knowing the application. [22, 23] are discussing this for multi media applications.

Security of mobile devices and related protocols such as 802.11b have attracted significant research effort. So, many proposals have been made to protect data during transmission etc. But to the best of our knowledge, protection means against network based attacks that attempt to hijack the mobile device, to render it useless or to steal information out of its memory have not been researched. There are several solutions which provide firewall protection on the IP layer. For mobile devices running under Linux, packet filters, such as netfilter [1] or nf-HiPAC [2], can be used. For PDAs running under MS Windows the first commercial packet filters by Bluefire [5] and Trust Digital [6, 7] are available.

But since Web Services are becoming more and more widespread and commonly used, these mechanisms are no longer sufficient to protect mobile devices against attacks due to the fact that the related SOAP calls are tunnelling the IP packet filters. Potential attacks are SQL code injection to get access to confidential data and recursive payload of XML messages which consume the whole memory while being processed and thus render the mobile device useless [16]. While the first may not be very probable in the near future the latter is already feasible. A proper application of Web service security features as given in the WS-Security framework [8, 9], and Security Assertion Markup Language (SAML) [14] helps to protect the mobile device against those attacks. But up to now there is no Web Service firewall solution for handheld devices available. Reactivity [10] and Forumsystems [11] are providing such solutions for server class systems.

In this paper we are focusing on cross layer interaction, and we are taking advantage of the fact that we are using the layer interaction for a specific application, i.e. firewall functionality for mobile devices. In contrast to most other approaches, we are introducing a specialized component, which is responsible to deduce the next step based on the data provided by individual layers.

3. Cross Layer Interaction to Reduce Firewall Effort

3.1 Basic Idea

In order to protect computers within company's network several specialized routers/computers are deployed. In such a network configuration a router does the IP packet filtering and on a separate computer, i.e. on a bastion, TCP- and application level gateways are executed. With such architecture, it is possible to separate the external and internal world physically as well as logically. In contrast to this, the mobile device is exposed directly to attackers. Thus, it has to execute all protection mechanisms itself.

Providing a significant level of security enforces the use of packet filtering, application level gateways or similar approaches - and exhaustive use of cryptographic means. Due to the still limited resources of mobile devices, all these means have to be applied as efficient as possible.

From a security point of view as well as from efficiency point of view it is highly desirable to establish a defence line as close as possible to the network interface. This means that malicious packets should be identified and blocked at the lowest possible layer. This has the following benefits:

- All higher layers are protected from malicious packets.
- The mobile device does not spent battery and calculation power to process malicious packets at higher layers.

But packet filtering firewalls cannot detect whether an incoming packet is malicious or not, they just can check if the packet belongs to an existing connection and if the source address is already blocked etc. For example http packets usually tunnel IP firewalls. Thus, application level gateways that do content inspection are needed to secure mobile devices. This task is very time consuming and should be done only if it is really necessary. In other words, it should be avoided whenever possible.

So in order to reduce processing effort at the application layer, i.e. avoiding content inspection of malicious packets, we propose to enable application layer gateways to update packet filters at lower layers. If the application layer gateway detects a malicious packet its source address is added to IP layer filtering rules. In order to verify the feasibility and the potential performance benefit of this idea, we implemented an application layer gateway on a PDA and realized the application to IP layer communication. The implementation details are given in the next sub section.

3.2 Implementation

In order to evaluate the potential gain of our cross layer approach we realized an architecture running an IP layer firewall and an application layer gateway. We enabled the application layer gateway to extend the IP filtering table with new IP addresses, after detecting malicious packets.

We are using two kinds of IP packet filters namely nf-HiPAC and Netfilter (IPtables) in order to measure the performance of IP layer packet filtering. Here we investigated how the number of rules influences the performance of the packet filtering. This is important since due to our propagation strategy the number of filtering rules will increase constantly. As application level gateway we are using Xerces-c [12] to validate incoming XML packets, here SOAP calls or answers, and XML security for data integrity check, authentication etc. The cryptographic means as well as a rigorous verification of the SOAP packets against a strictly defined XML schema provides reasonable first step towards application level firewall functionality.

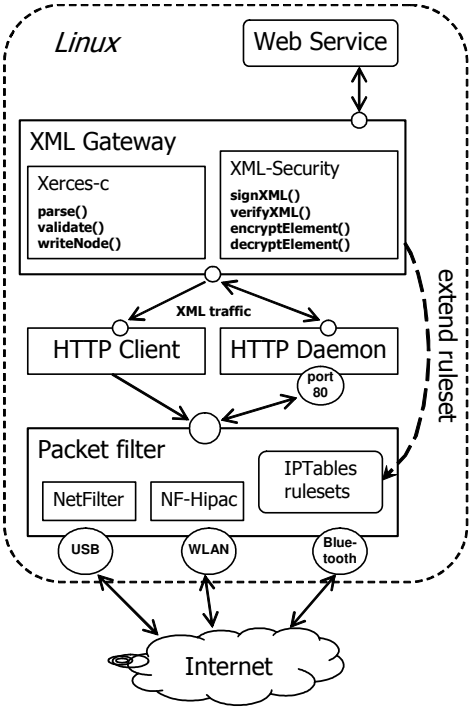

Figure 1: Current implementation of FPM, displaying used software modules and cross layer interaction

We developed our example Web Service using the gSOAP framework [15] that provides access to sockets also on application level. Thus, we did not include the address resolution in our first implementation. But compared to potential gain and the quite high effort to detect malicious packets on the application level, the additional effort for address resolution can be neglected. Figure 1 shows the current realization of the system, running on an HP iPAQ h5550.

4. Measurements

Measurement set-up Figure 2 shows the network we used to run the measurements. As mobile device we used an HP iPAQ Pocket PC h5550 with a 400MHz Xscale Processor, 128 Mbyte SDRAM, 48 Mbyte flash ROM and an integrated 802.11b modem. The iPAQ is running the Familiar Linux distribution version 0.8 [13]. As communication partner we used an Acer 290 notebook, running SUSE Linux version 9.1. The wireless connection was set up using a D-Link DI-624 router providing an 802.11g wireless access point and four 100Mbit Ethernet interfaces. The wired connection was realized with USB1.1 cable attached to the notebook and to the iPAQ. We used the USB connection for the following reasons: first, it ensures that the computational load on the PDA is not limited by the available bandwidth and second, it is used to guarantee reproducible measurement results, due to the fact that no network errors occur in a wired connection1.

On top of our sample network we realized a very simple Web Service client that does nothing else than accepting incoming "answers" from a Web service provider. In addition we realized a simple Web service provider, which accepts a request and sends back an answer that has a size of 1kbyte. Thus, the execution of the Web Service does not influence the measurement results. Our Web service client and our Web service provider have been realized using the gSOAP framework [15], and both can be executed on the PDA. In subsection 4.2 we present measurements done with the Web Service client running on the PDA. In this case we present measurements for different packet sizes, which may result from different requests to the service provider side. In subsection 4.3 we present measurements of the Web service provider running on the PDA. Here the size of the incoming packets is no longer of interest since we assume that normally the size of request is quite small, but a lot of request is sent to the service provider which will cause the major part of the effort at the provider side.

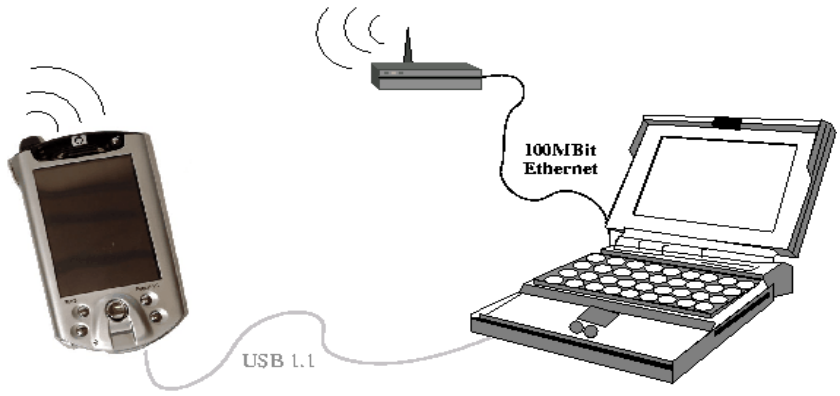

Figure 2: Measurement set-up

4.1 IP layer packet filtering

Filtering of packets at the IP layer is the first step to secure a handheld device. This is why we first measured the performance of IP packet filter approaches as a first simple step towards securing a handheld device against malicious communication partners. We altered the source address of the incoming packets on a per packet basis in order to ensure that all rules had to be checked before an incoming packet was recognized as malicious, i.e. our measurements presented later show the performance in a worst case scenario. In these experiments all IP packets had the maximal size but were not fragmented.

Our measurement results show that the additional CPU load is about ten per cent as long as the number of rules sets2 applied is less than 200, see figure 3. Please note that the CPU load is about 20 per cent even without running any security function when a wireless connection is used, so additional 10 per cent of CPU usage seems to be an acceptable price for increasing the security of the mobile device. The situation changes when the number of rule sets is growing. But even then, more than 1000 rule sets have to be applied before the CPU load is about 50 per cent. So, as long as no really exhaustive use of rule sets is made packet filtering can be applied

[1] Due to space limitations, we do not provide results measured while USB was used. In addition we do not discuss the impact of the computational load of the PDA on usable bandwidth.

[2] Each rule set contains 2 outgoing (stateless + stateful) rules and 2 incoming rules

without turning the mobile device useless. If nf-HiPAC is used the additional effort for filtering is even less than 5 per cent, independent of the size of the rule set3.

Our measurement results clearly indicate that IP layer filtering has some impact on the performance of the mobile device. But with a reasonable number of rule sets, it is feasible to use IP packet filtering.

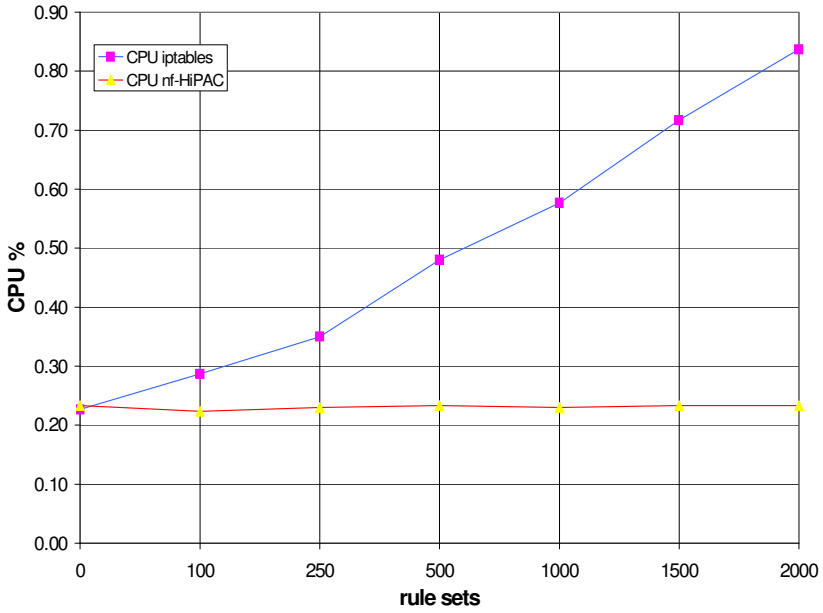

Figure 3: CPU usage in per cent vs. size of the applied rules set for IP layer packet filtering using of IP tables and nf-HiPAC; measurements done on HP IPAQ h5550; using a wireless connection via 802.11b

4.2 Web Service Gateway

As second experiment we measured the performance of an application level gateway that is securing a web service. Here we investigated the performance of several XML security issues such as encryption and digital signatures. In this experiment we used Web Service request with a size of 1Kbyte, 10Kbytes, 100Kbytes and 1000Kbytes.

In order to ensure secure operation at the application layer a lot of different techniques have to be applied. First, a certain packet containing SOAP requests has to be validated against the XML scheme of the corresponding Web Service. In addition, the XML security features such as en-/decryption and digital signatures have to be taken into account. The measurements discussed in this subsection present the client side of our simple Web service and have been measured on an iPAQ H5550.

In order to verify the computational load caused by the different security functions we have measured the following configurations:

- **Os**: Original set-up (mini-httpd and http) without any security functions used as benchmark and to calculate the overhead resulting from the security function used in other configurations.
- **Bc**: basic configuration XML packets are parsed and optionally serialized
- **BcV**: like Bc including the following addition: validation of incoming packets against corresponding XML scheme
- **BcS**: like Bc including the following additions: signature generation and verification (signXML at server, verifyXML at client)
- **BcE**: like Bc including the following additions: En-/decryption of XML packets (encryptXML at server, decryptXML at client)
- **BcES**: like BcE including the following additions: signature generation/verification

Figure 4 shows the percentage of CPU usage for the application level gateway configurations explained above. For each configuration we measured requests with the size of 1Kbyte, 10Kbytes, 100Kbytes and 1000Kbytes.

[3] The independence of nf-HiPAC of the size of the applied rule set was already measured for PCs and presented in [17]

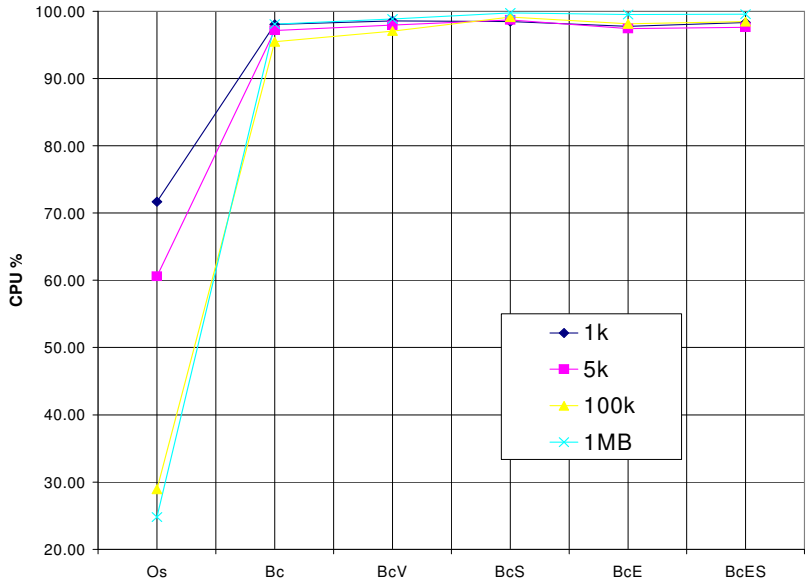

Figure 4: CPU load of an iPAQ running our Web service client with diverse security settings applied for incoming Web service answers

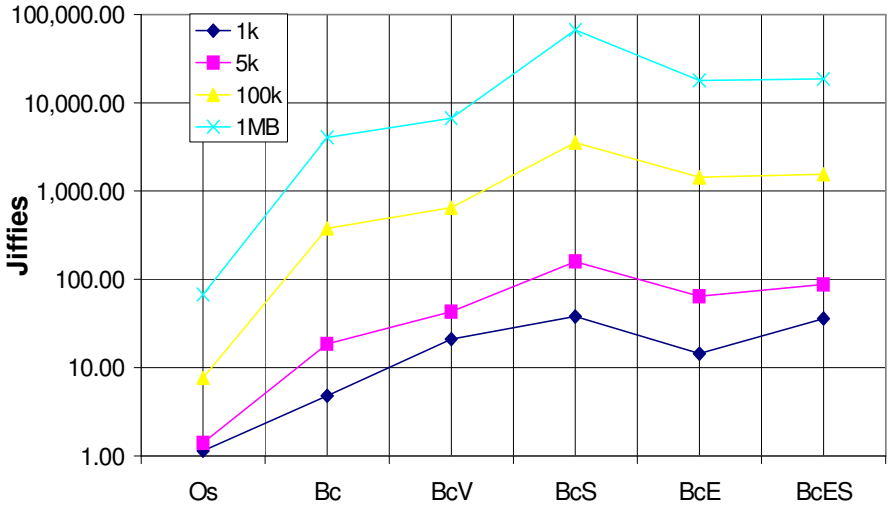

Figure 5: CPU load in jiffies4 for diverse settings of our application level gateway running on an iPAQ h5550; measurements displayed for the following packet sizes: 1000Kbyte, 100 Kbyte, 5Kbyte and 1Kbyte requests

The measurements depicted in Figure 4 show clearly that the additional security functions lead to a significantly increased CPU load. It is even increased significantly if the incoming packets are only parsed and validated. Figure 5 shows the load distribution in jiffies per scenario and size of requests in more detail. Comparing figure 4 and figure 5 shows that despite the CPU load in per cent is more or less equivalent for all sizes of requests, the absolute CPU load in scenarios with small packet sizes is relatively low. In addition, it can be seen that especially the XML security functions en-/decryption and the application of digital signatures increase the CPU load. The peak load in the BcS configuration results from the effort inhibited by the need to bring the request/answer packets in the canonical format, which is required to ensure proper signature verification at the receiver side. The effort for this operation is dramatically reduced if the packet is encrypted (see fig. 5 BcES scenario) since the encrypted data is in canonical form already, which ensures that less tokens have to be evaluated.

[4] Jiffies denote the number of hardware clock ticks that were counted in a certain time interval; the number of jiffies and the CPU usage are stored at /proc/uptime and /proc/stat by the Linux system, whenever running. Thus, we did not need to include measurement or profiling instruction, so that our measurements are not influenced by any measurement procedure.

Depending on the Web Service and its configuration, at least parsing and validating have to be done in order to ensure proper operation of the running Web service client or service. Thus, the additional overhead caused by the application level gateway is quite low, i.e., the difference of the CPU load between the configuration BcV and BcS is less than 20 per cent. The performance penalty caused by the security functions can be minimized if the application level gateway passes the reference to the parsed tree and the result of the validation to the Web Service. So, these functions have to be executed only once.

4.3 Cross layer interaction

The effort to secure a handheld device is significant, as already shown earlier in this section. In order to reduce the total effort we propose to realize a vertical cross layer interaction between application level gateways doing content inspection and firewall solutions on lower layers. In order to evaluate the potential performance gain of our approach we measured the following scenarios:

- **Original**: no malicious packets
- **Original2**: all packets are malicious, and the detection is done at the application level
- **Original3**: like original2 but only 50 per cent of the packets are malicious
- **iptables1**: 50 per cent of the packets are malicious; the first malicious packet of a certain source is detected at the application level, then the IP netfilter is updated and all following malicious packets from the same source are detected and dropped at the IP layer
- **ipables2**: like iptables1, but all packets are malicious.

Figure 6 shows the CPU load for all scenarios. Comparing the scenarios Original2 and iptables1 clearly indicates that updating the IP netfilter rule sets helps to reduce the CPU load significantly. In our example the CPU load is reduced by about 10 per cent, in case that 50 per cent of the incoming packets are malicious. If the number of malicious requests is increased the benefit resulting from our approach increases too. In addition to the reduced CPU load our approach enables the Web service to serve more requests within the same time interval. The scenarios original3 and IPtables1 have been executed under the same conditions, i.e., half of the incoming requests have been malicious. Due to the application of our cross layer approach about 300 additional requests more have been processed in the latter scenario. Please note that the benefit resulting form our approach is due to the fact that all incoming packets in scenario iptables2 are malicious, which means they are processed only once by the application level gateway, and afterwards they are blocked by the IP layer firewall.

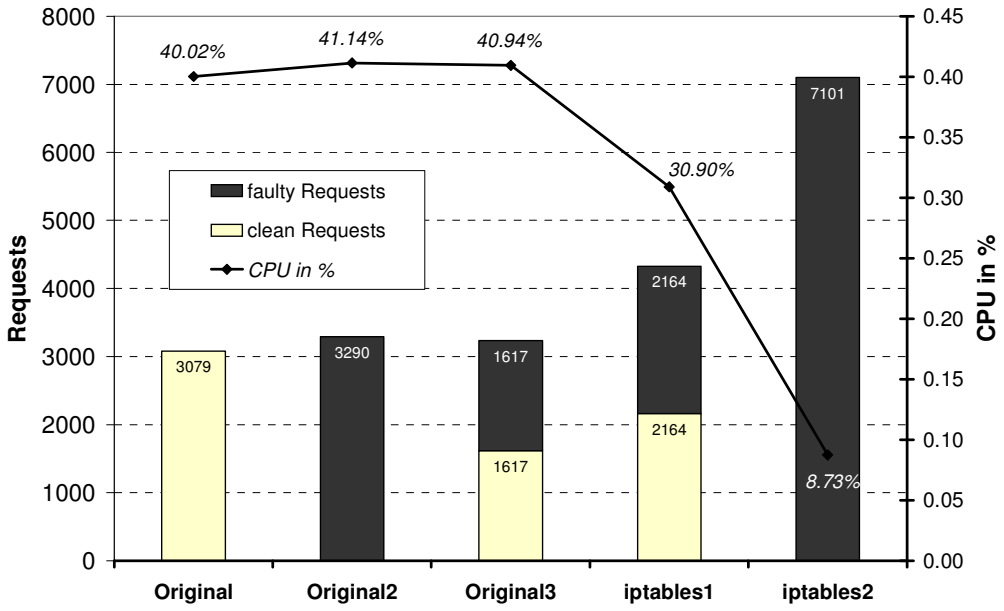

Figure 6: CPU load caused by Web service provider with and without using cross layer interaction in scenarios with different percentage of malicious request; measurements done on an iPAQ h5550

5. Extending Cross Layer Interaction to Lower Layers

Our measurements clearly indicate that pushing firewall effort to lower layer is beneficial. It reduces the CPU usage and helps to increase throughput of other connections and battery lifetime. So, the next step is to verify whether we can increase the performance gain by pushing the firewall effort further down the protocol stack, i.e. down to the MAC layer, or more correct before the MAC layer.

In order to evaluate the improvement caused by the MAC firewall, we compared the performance of it against an IP firewall. Concretely, we measured throughput and CPU usage of the IPtables firewall as it is described in the previous section against a firewall that is based on the EBtables tool. EBtables [24] is a tool that is meant for bridging packets on the Ethernet layer. Configured as bridging router (brouter) it allows to filter the packets before entering the actual network stack. The brouter decides, following a list of rules, whether an incoming packet is directly put in the network stack or should enter the bridge. We configured the actual bridge so that it ultimately drops every packet. The scheme is shown in Figure 7 In this configuration it is an adequate MAC layer firewall.

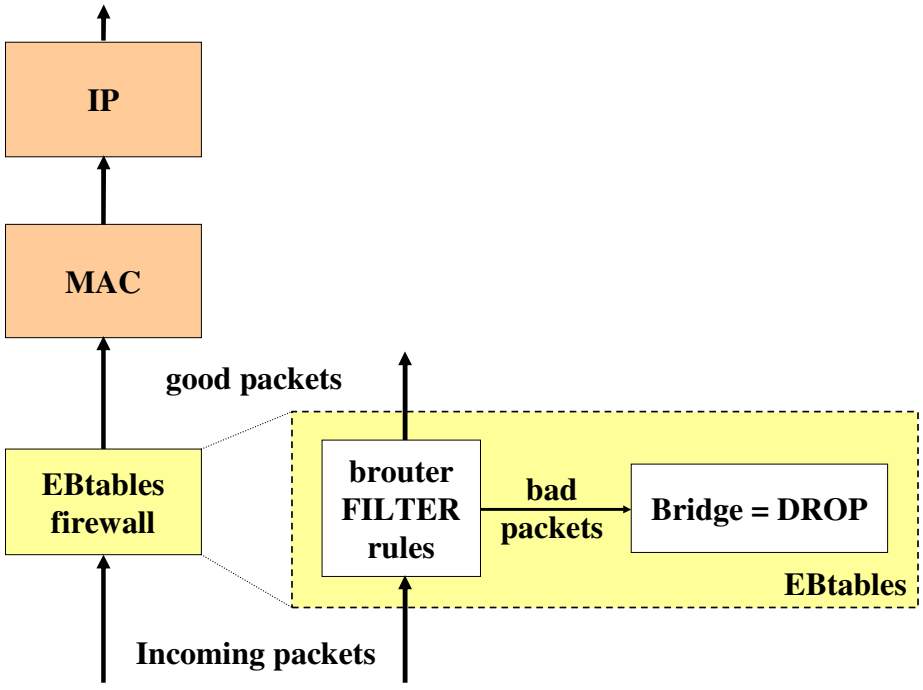

Figure 7: The idea of the MAC firewall

The source code of EBtables bases mainly on code of IPtables. For our tests this is very beneficial, since it promises very comparable results. In order to isolate the measurements on the pure network processing and to suppress protocol overhead we used UDP instead of TCP packets, in these tests. We performed the tests with small UDP packets (50 bytes) and large UDP packets (1460 bytes) sent via WLAN on firewall configurations with 100, 500, and 1000 firewall rules. Every packet has to pass each rule before it finally is accepted or dropped. The large packets correspond to the usual TCP data transfer packets, while the small packets are kind of worst case test for firewalls.

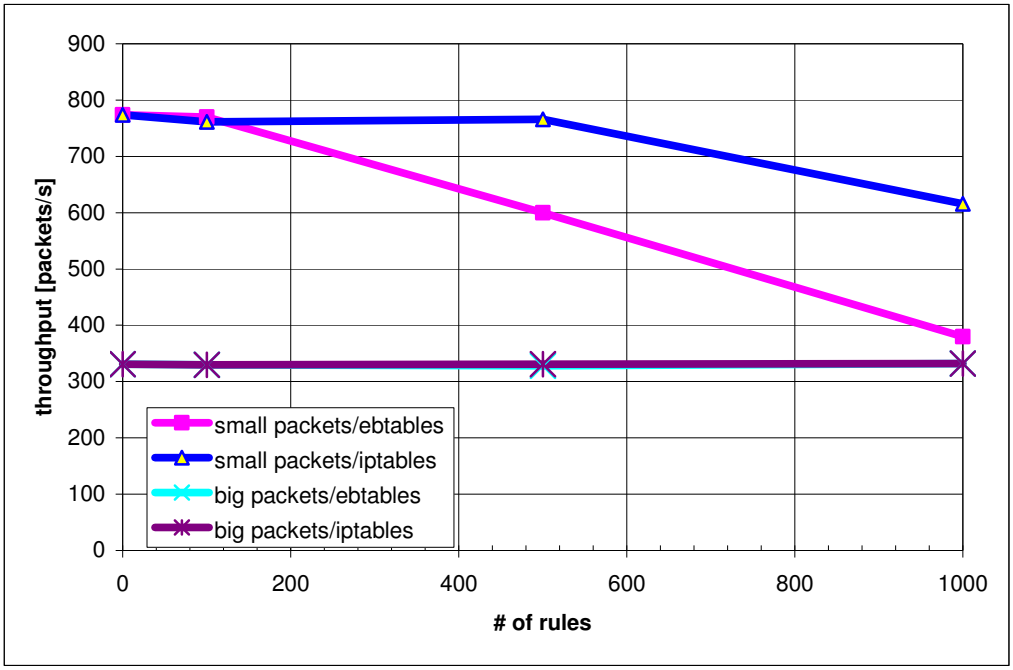

Figure 8: The throughput of the firewall solution depending on the number of rules and packet size

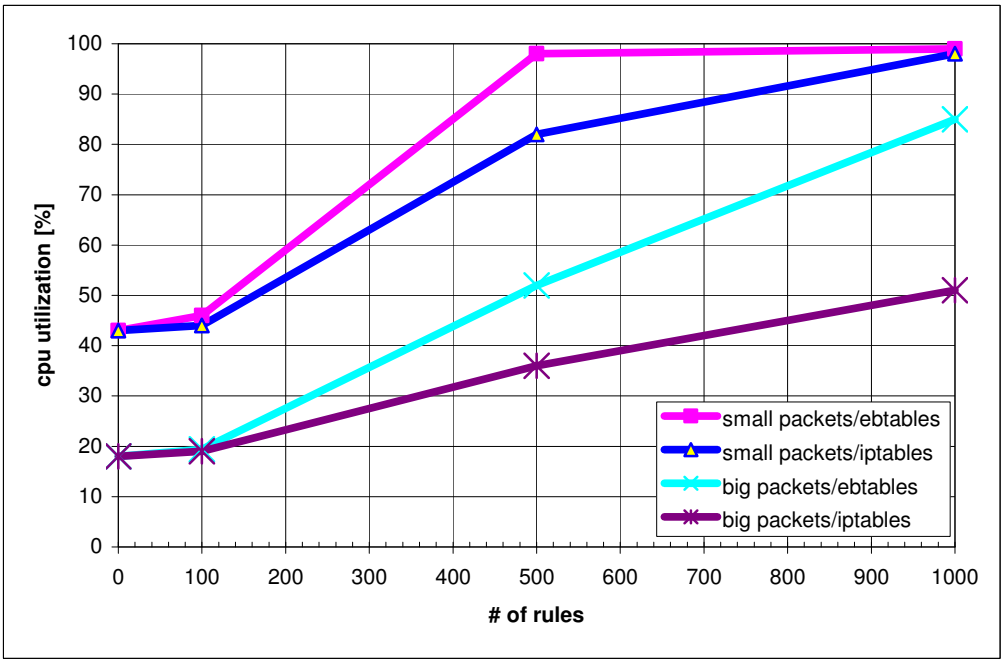

Figure 9: The CPU utilization depending on the number of rules and packet size

Figure 8 shows the throughput measured in packets per second. The throughput for big packets is not affected by the firewall rules. This is because for such big packets the bottleneck is not the header processing but transmitting and copying the packet with the payload. The small packet throughput is much more affected by the header processing and thus by the number of firewall rules. The 1000 rules test processes 60% less packets with EBtables than without firewall. For IPtables the decrease is still about 40%.

The CPU utilization that is presented in Figure 9 shows the reason for the throughput data. The throughput for the big packets does not go down because the corresponding CPU usage is always less than 100%. However, the usage rises with increasing number of rules. The tests with small packets and many rules are clipping at the 100% CPU utilization. This is the reason why the packet throughput decreases. The CPU usage for the EBtables

tests is significantly higher than the corresponding IPtables tests. This is why the EBtables throughput decreases faster than the one of IPtables.

It is a bit surprising that the performance of EBtables is so much worse than IPtables. But taking a look in the network stack can explain it. The MAC firewall gets the packet, extracts MAC header, the network protocol and if necessary the IP header. Then the filtering is performed. If the packet is accepted it will be forwarded to the network stack where again the headers are read and evaluated. Beside the multiply executed header processing other reasons are responsible for the worse performance:

- A MAC firewall has to access and extract more data. While IPtables only has to evaluate the IP related content only, EBtables need to handle the whole packet. This means that at least the MAC addresses and network protocol must be evaluated additionally.
- In the IPtables implementation the IP addresses and ports are checked very early in a fast path. It can be done because IPtables knows it gets IP packets only. In EBtables the check for IP parameter is part of a special module that must be explicitly called.
- The MAC firewall has to process more packets than the IP based one, because the IP packets are just a subset of all packets that pass the MAC firewall. Additionally it processes packets for other hosts before their destination is checked in the MAC layer.

In order evaluate filtering overhead per module we performed another simple test: The firewalls were configured with no rules but the policy to accept or drop the packets. A packet must pass all preceding layers, i.e. when it is dropped by the IP layer it was accepted by both firewalls and MAC layer. Indeed, it is no a practical setup, but it shows the filtering overhead. We sent small UDP packets as fast as possible i.e. with 780 Pk/s. The measured CPU utilizations are shown in Table 1.

Table 1: The CPU utilization while packed dropped depending on the layer

Packet dropped at	EBtables	MAC layer	IPtables	IP layer
Reason	Policy	bad MAC addr.	Policy	bad IP address
CPU utilization	39.5%	40.0%	42.5%	43.0%

The results presented in Table 1 strongly indicate that the filtering effort is relatively low compared the protocol processing effort in the base ban and physical layer. Therefore we propose an interleaving of firewall and protocol processing at the border of the MAC layer in the next section.

5.1 Cross layer communication

We propose to use an extended version of the Firewall Management Plane (FMP). The FMP gathers data about malicious or at least suspicious packets. As soon as the suspicion is proven, it updates the filtering rules available on lower layers. Whether a packet flow is considered to be malicious depends on the policy deployed at the mobile device. A single packet which is, e.g. not well formed or which could not be validated may probably not be considered as an attack. But it should be recorded that such a packet was received together with its source address. If a sequence of suspicious packets coming from the same source is detected, it might still be caused by network errors not detected at lower layers, but it may also be a kind of attack. Thus, the source address should be marked as malicious and the firewall entries of the lower layers have to be updated. The maximal number of suspicious packets that are tolerated from a certain source, before this source is marked as suspicious or malicious, is also specified in the policy file. So, the policy file reflects how cautious the user is.

If a malicious source is detected, both IP and MAC address are stored in the table of suspicious packets together with the destination IP port and the rank of the suspicion taken from the policy file. If a similar packet appears again the total rank is increased and if it reaches the threshold all new packets from the source will be dropped by the firewall. Indeed, the blocking of MAC addresses must be done very carefully to avoid the case where the firewall drops all packets from a gateway or an access point that only routes them. If the packets are routed, the filtering must be applied over IP data. However, in mobile environments, in particular in ad-hoc networks, the devices are much more often directly connected to new or unknown peers, so that the filtering of MAC addresses is very reasonable.

In order to propagate information about malicious sources from higher layers to lower layers the FPM needs to resolve, e.g., URLs in order to get the corresponding IP address. This can be done using the already available

mechanisms. Figure 10 shows the overall architecture and the interaction between the Firewall Management Plane and the firewall solutions on different protocol layers.

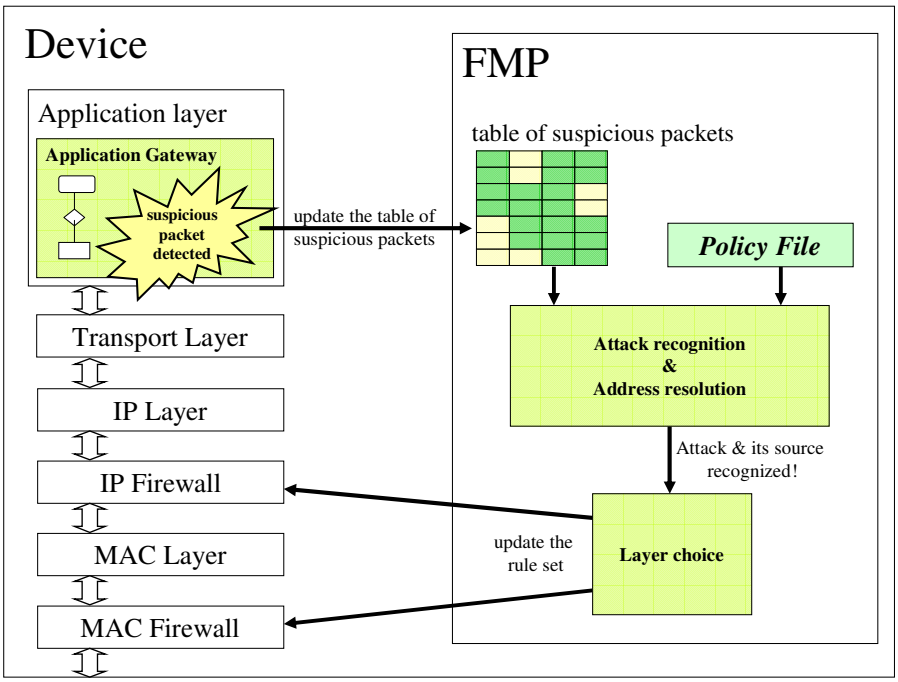

Figure 10: Schematic view of the proposed cross layer architecture for a firewall management plane including content inspection on the application layer

5.2 Integrated Layer Firewall Processing

In order to improve the benefits resulting from early filtering of malicious packets we propose to allow interaction between the filtering firewalls and the base band processing, and the firewalls and the network protocol stack.

Figure 11 shows the envisioned architecture. Here the header processing functionality of the MAC and the IP layer is separated from the remaining functionality and executed immediately after the base band has completely processed the MAC header. In the first step the MAC header is inspected. If the packet is malicious or has a different MAC address as destination address a STOP signal is sent to the base band process and the processing of the payload is aborted.

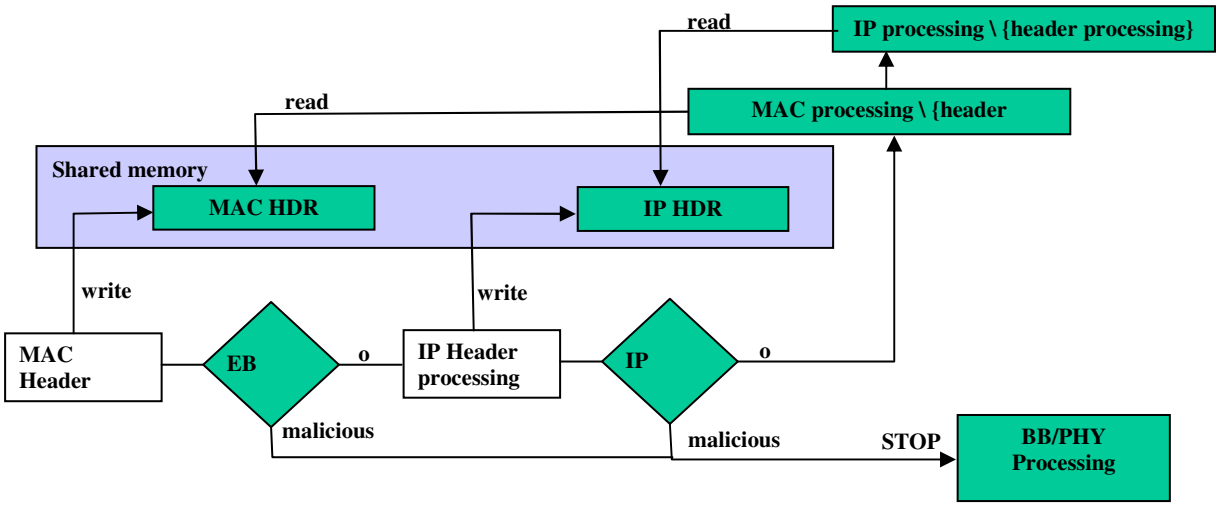

Figure 11: The architecture of the proposed Integrated Layer firewall solution

Given the fact that for short UDP packet nearly 40 per cent of the processing power is consumed before the MAC processing is started, this interweaving of lower layer with filtering approaches can lead to a significant reduction of the CPU usage. The actual benefit depends strongly on the situation; if no attack is running nothing will be saved.

We currently have no own measurements to proof the benefit of our idea. But in [18] the authors propose early MAC address recognition and stated that the MAC header is just five to ten per cent of the complete MAC packet. Depending on the current situation (number of mobile devices within the same hot spot) the potential benefit varies between 0 (no other device) and 75 per cent. With our idea the probability to save energy Is increased due to the fact that in addition to packets with different MAC address, malicious packets that in the approach of [18] would be processed are dropped before the processing really starts.

The interaction of the firewall and the network processing layers reduces the effort for accepted packets. In the standard approach, the firewall evaluates the header of the packet, but the only information that is forwarded to the network stack is that the packet was dropped or accepted. The header processing must be performed again by MAC and IP layer. We propose that in case the packet is correct the already processed MAC header is stored in a memory area, which can also be accessed by the remaining MAC procedures. The IP header is processed in the same way. This architecture should reduce the performance penalty observed in the EBtables tests substantially.

6. Conclusions and Outlook

In this paper we have investigated the performance of IP layer firewalls as well as the performance of application level gateways for Web Services on handheld devices. Our measurements indicate that it is feasible to use firewalls on mobile devices. The CPU load caused by IP layer filtering is less than 10 per cent, even if about 200 rule sets are applied. We have proposed to use a cross layer interaction to reduce the performance penalty that results from the use of application level gateways on handheld devices. Our experiments show that layer interaction can reduce the CPU load by about 30 per cent. The actual CPU load reduction depends on the number of malicious packets as well as on the security means applied.

Pursuing a policy of removing malicious packets as close as possible to the network interface we moved the firewall to the MAC layer. These experiments could not show any further performance improvements. This is caused by implementation issues but also by the fact that the packet headers that already have been evaluated by the firewall are processed again on higher layers. In order to eliminate this processing overhead we proposed an integrated layer firewall processing. Data and conclusions that have been made by the firewall are shared with the actual network processing layers. Thus, the extraction and evaluation of IP or TCP data even before entering MAC layer can be beneficial. A further improvement could be the proposed control of the base band processing by the firewalls. After processing the header of a malicious packet, the firewall can send a signal to the base band that stops the reception of the packet's payload.

The latter approach has not been implemented and proven yet. It will be one of our next research steps to do it and to measure the resulting performance implications. Furthermore, we will evaluate the power consumption that stems from the security functions running on a handheld device. We will also evaluate the impact of hardware accelerators for cipher mechanisms on the performance relations.

Acknowledgements

This work was partially funded by the German Ministry of Education and Research under grant 01AK060B.

References

[1] Netfilter/iptables Project Homepage. www.netfilter.org
[2] nf-HiPAC: High Performance Firewall for Linux Netfilter. http://www.hipac.org
[3] Extensible Markup Language (XML) 1.0 (Third Edition). http://www.w3.org/TR/2004/REC-xml-20040204
[4] http://webservices.xml.com
[5] Wireless Security Software for Handheld Mobile Devices from Bluefire Security Technologies. http://www.bluefiresecurity.com/
[6] Trust Digital - Solutions - TRUST Mobile Device Applications, http://www.trustdigital.com
[7] Security Basics for PDAs and Handheld PCs, http://www.smallbusinesscomputing.com/webmaster/article.php/10732_3400641_2

[8] Web Services Security (WS-Security), http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
[9] XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core/
[10] Reactivity: The Secure Web Services Deployment System. http://www.reactivity.com/
[11] Forum Systems, Inc. - The Leader In Web Services Security. http://www.forumsystems.com
[12] XML-Security-C. http://xml-security-c.sourceforge.net
[13] Handhelds.org - Open Source Operating Systems for Handheld Devices. www.handhelds.org
[14] OASIS, Security Assertion Markup Language (SAML) V2.0 available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20
[15] Robert van Engelen, gSOAP 2.7.2 User Guide, available at: http://gsoap2.sourceforge
[16] Forum Systems: Anatomy of a Web Services Attack: A Guide to Threats and Preventive Countermeasures, 2004 available at: http://forumsystems.com/papers/Anatomy_of_Attack_wp.pdf
[17] Michael Bellovin: nf-HiPAC High Performance Packet Classification High Performance Packet Classification for Linux Netfilter, 2005, available at: http://www.hipac.org/documentation/nf-hipac-nfws2005.pdf
[18] M. Methfessel, H. Frankenfeldt, K.F. Dombrowski, and R. Kraemer Optimizing the Downlink for Mobile Wireless Devices , N. C. Beaulieu and L. Hesselink (Eds.) Proceeding of Wireless and Optical Communications (WOC), 2002
[19] M. Methfessel, et. al: *Vertical Optimization of Data Transmission for Mobile Wireless Terminals*. IEEE Wireless Communications, 2002.
[20] P. Langendörfer, et al.: *Shielding TCP from Wireless Link Errors: Retransmission Effort and Fragmentation.* The Journal of Supercomputing, Vol. 23, (3), 245-260, 2002.
[21] Gustavo Carneiro et al.: *Cross layer design in 4G Wireless terminals*, IEEE Wireless Communications, Vol 11, No 2, 2004 Island, May 2003.
[22] Radu Cornea, Shivajit Mohapatra, Nikil Dutt, Alex Nicolau, Nalini Venkatasubramanian, "*Interactive Managing Cross-Layer Constraints for Mobile Multimedia*", IEEE Workshop on Constraint-Aware Embedded Software(RTSS-2003) — Cancun, Mexico, December, 2003.
[23] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, *Design and Evaluation of A Cross-Layer Adaptation Framework for Mobile Multimedia Systems*, in Proc. of SPIE/ACM Multimedia Computing and Networking Conference (MMCN'03), Santa Clara, CA, January, 2003.
[24] EBTables project homepage: http://ebtables.sourceforge.net