FFI-rapport 2009/00747

Distributed certificate validation in MANET

Anders Fongen

Norwegian Defence Research Establishment (FFI)

19th May 2009

FFI-rapport 2009/00747

1086

P: ISBN 978-82-464-1580-2
E. ISBN 978-82-464-1581-9

Keywords

MANET

XKMS

Sertifikater

Approved by

Anders Eggen	Project manager
Vidar S. Andersen	Director

English summary

The need for certificate management in mobile ad hoc networks (MANET) is the background for this report. A prototype implementation of a distributed certificate validation service is presented and evaluated.

The proposed design is based on an overlay network of proxy nodes offering certificate validation over the XKMS protocol. The proxy nodes employ cooperative caching in order to offer certificate validation even when the central validation authority is out of reach. The cache will also contribute to lower network traffic in the area around the validation authority.

The conclusion from the experimental evaluation shows that the availability of the validation service does increase as a result of the overlay proxy network, but is quite dependent on the mobility scenario in use. The observed traffic around the validation authority is reduced with a considerable margin.

Sammendrag

Denne rapporten har sin bakgrunn i behovet for sertifikatvalidering i MANET. En prototypisk implementasjon av en distribuert tjeneste for sertifikatvalidering blir testet og evaluert.

Den foreslåtte løsningen baserer seg på et overlay-nettverk av proxy-noder som tilbyr sertifikatvalidering med protokollen XKMS. Proxy-nodene samarbeider om å cache nylige valideringsresultater på en slik mate at validering kan skje også når den sentrale valideringstjeneren er utenfor forbindelse. Cachene vil også bidra til å redusere trafikken i den delen av nettverket som er nærmest valideringstjeneren.

Konklusjonene fra den eksperimentelle evalueringen viser at tilgjengeligheten av en valideringstjeneste blir forbedret av proxy-nettverket, men at dette er avhengig av mobilitetsscenariet som benyttes.

Nettverksbelastningen blir betydelig redusert, spesielt i området av nettverket rundt hovedtjeneren (root VA).

Contents

Preface

The FFI project 1086 named “secure pervasive SOA” investigates how SOA principles can be applied to a military information system. The openness of SOA raises lots of security concerns with regard to integrity, authenticity, confidentiality and access control of services and of the information being processed by these services.

These concerns are presently being addressed on a large scale by the industrial community, and several standards have been established on how the security mechanisms may be represented by XML constructs. The basic mechanisms for signing and encrypting information is in place as well as frameworks for authorization and certificate management.

In a military environment, these standards may represent a challenge for the part of the communication infrastructure which has poor connectivity and low bandwidth. The protocols designed for a stable and high-speed network become too costly both in terms of transport volume and the number of necessary protocol interactions.

These issues are addressed in this report. A design for a distributed certificate validation service is being presented, together with a comprehensive experiment which evaluates the efficiency of a design prototype.

1 Background and justification of research

Mobile nodes which communicate through wireless technology may form spontaneous and dynamic multi-hop networks often termed as MANETs. Due to the dynamic topology of MANETs the nodes often experience broken connection as the network is partitioned, i.e. divided into isolated "islands". When MANETs are using military radio equipment, the bandwidth of the radio links are often found to be smaller than what is offered by equipment for civilian use. The reason for this is the emphasis on low signature and strong protection found in military radio design.

1.1 The need for mobile middleware

Low bandwidth and episodic connectivity is therefore the properties that distinguish a MANET from stationary, managed networks. Tactical military communication systems, for which MANET technology is of great interest, must employ middleware and application services which are able to satisfy their requirements under MANET conditions.

Compression, replication and proxy services are techniques which may improve the conditions which characterize a MANET. *Compression* reduces the volume of data transported over a physical connection, but does not help when links are broken. *Replication* means that data is stored on several places, which increases the chance of finding a data item when the connectivity becomes limited. *Proxy services* means that application services are offered from several places in the network on behalf of a central/authoritative server. Proxies increase the availability of the service due to the distribution of service points. Both replication and proxies may reduce the transported data volume as well, since the path from a client to a replica/proxy may be shorter than the path to the central server.

1.2 The need for certificate validation

Before a node acts on a received message, it is sometimes necessary that the source and the integrity of the message is verified. A digital signature can be used to verify the integrity of the message and the identity of the signatory, provided that the receiver is in possession of the signatory's correct public key. This is where *public key certificates* come to rescue: the certificate binds the identity of the signatory to his/her public key through the signature of a "trusted third party". The certificate is a public document which can be stored and distributed everywhere, but its *validity* may be revoked if, for some reason, the public key should no longer be used. The reason for a revocation may be a secret key compromise, the death or reassignment of the key holder etc.

Consequently, there is a need to validate a certificate at regular intervals in order to control the risk for incorrect validation, i.e. accepting a certificate even though the certificate is revoked. The risk can never be completely eliminated, since there is an inevitable non-zero latency from the moment a certificate is administratively declared as revoked and the moment from which the certificate is never accepted by any relying party (i.e. the actor that validates a certificate). This period is termed *revocation latency* and is determined by the chosen distribution mechanism for revocation information.

There exists a well defined architecture for the generation and deployment of key pairs, and for the dissemination of certificates and revocation information. These services are commonly known under the name *Public Key Infrastructure* (PKI). Note that a PKI does not offer validation services, but a set of simple services for use during certificate validation:

- Retrieval service for certificates
- Retrieval service for certificate revocations lists (CRLs)
- Query service for certificate revocation status

1.3 Client-based or authority-based validation

A consequence of the rather simple service level offered by a PKI (retrieval services, not validation services) is that the validating client becomes directly responsible for the steps involved in a validation operation. The steps necessary to validate a certificate include (but is not limited to):

1. Verify that the certificate is authorized for this usage (encryption, signing etc.)
2. Check that the certificate path length has not exceeded its maximum value
3. Ensure that current time is within the certificate's validity period
4. Look for this certificate on the revocation lists, fetch more recent revocation lists if necessary
5. Retrieve the issuer's certificate
6. Verify that the certificate is correctly signed by the issuer's private key
7. If issuer is not the trust anchor[1], repeat from step 1 with issuer's certificate.

A common variation to this list of tasks is to replace the use of revocations list with a query to the previously mentioned certificate status service which responds with the revocation status of a specific certificate.

[1]The term "trust anchor" is used in RFC 3280 to denote the owner of a certificate who is trusted per definition. In a PKI, your certificate issuer (CA) is likely to be your trust anchor.

This list of tasks indicates that certificate validation involves frequent access to centrally located servers and requires highly available network resources. Certificate validation also involves signature verifications which are computationally expensive operations.

Another approach, as found in the SCVP [7] and XKMS [8] recommendations, delegates the retrieval and validation of certificates to a *validation authority* (VA). The VA is an application server which relieves the client from a large software installation footprint (essential in lightweight nodes) and several computing tasks. The VAs hide their implementation behind a service interface and can employ any implementation strategy. The distribution of CRLs to clients is no longer necessary, since certificate revocation takes place behind the service interface.

1.4 Distributed Validation Authority

On lower tactical levels of a military network, bandwidth is a scarce resource, and the network links are frequently broken. In this environment, a certificate validation service which depends on reliable connectivity to a single set of services will not suffice. A better approach is to offer validation services from several endpoints in the network in order to increase the probability for at least one reachable endpoint.

The approach of a validation authority is interesting for this purpose, since the client's trust in the validation authority is based on a response with a *digital signature*[2]. The response from the VA is a signed document, which can be shared in any manner between all those who have trust in this signature (i.e. trust the VA). This report will use the term *Proof of Validation* (POV) to denote a timestamped document describing the validity status of a certificate, signed by a trusted validation authority.

It is possible to create a *cooperative caching* arrangement of VA proxy servers, which exchange and distribute POVs issued by the "root" VA, and which offers validation services based on sharing of existing POVs.

1.5 Outline of the report

The design and evaluation of a distributed certificate validation system based on cooperative caching of POVs is the subject of this report. The purpose of the effort is to study possible solutions to the need for certificate validation in wireless tactical networks. The remainder of the report is organized as follows: Chapter 2 presents the design of the distributed validation service, Chapter 3 presents a few implementation issues, and Chapter 4 describes the design of the evaluation experiment and the observations made. A discussion and analysis of the observations is also a part of this chapter. Chapter 5 concludes the report and Chapter 6 provides a list of acronyms used.

[2]The response may also be protected by an authenticated session, which is disregarded in this report for reasons of decoupling between transport and application protocols

1.6 Intended audience and required background

The report is written from a technical perspective and is intended for the reader who is well informed and experienced in distributed design and programming. The design builds to a large extent on the use of XML, Web Services, XKMS [8] etc. of which the reader should be somewhat informed. The reader should also be aware of the author's publications on XKMS [3], distributed certificate validation [5] and MANET experimentation [4][6]

1.7 Deliverables from this research

Apart from the publications shown on the list of references, the research efforts presented in this report have produced the following deliverables:

1. An XKMS server written in Java
2. An XKMS proxy node control program written in Java
3. A modified OLSR executive for integrated peer/service discovery
4. A set of software modules used to build a MANET emulation testbed
5. A Hierarchical Group Mobility Model (HGM) written in Java, used to control the mobility scenario during emulation

2 Description of design

This chapter will provide a description of the design of a distributed certificate validation service. The design has been presented on MILCOM 2008 [5]. The chosen protocol for the validation service has been XKMS.

2.1 Proxy overlay network

The central service point for certificate validation normally present in an XKMS implementation is replaced with an *overlay network of proxy nodes*. The term "proxy node" refers to a MANET node which has been equipped with the XKMS proxy software. These proxy nodes all behave like normal XKMS nodes in the sense that they implement the same service interface. To the client, "real" XKMS servers and XKMS proxy nodes are indistinguishable.

The proxy nodes are cooperating and exchange their cached POVs in an "on-demand" manner. In order for the nodes to cooperate they form an *overlay network* (also called *proxy overlay*) with a topology laid on top of the actual network network topology. The overlay nodes consist of a subset of the MANET nodes, and the connections between overlay nodes depend on actual network routes (paths) in the underlying MANET, see Figure 2.1.

2.2 Cache management and forward processing

Each proxy node keeps a cache of recent validation results (POVs). When a client sends a validation request to an XKMS proxy node, the service program in the node will look in its cache for recent results regarding this particular certificate. If a relevant and recent result exists, it will become the result of the operation and returned to the client. Otherwise, the request is forwarded to another proxy node according to a forwarding algorithm. During the forwarding operation the proxy node (A) becomes the client of another proxy node (B), and the result of the request to B is stored in A's cache before returned to the client (which may be a proxy node acting on behalf of someone else), as shown in Figure 2.2

The forwarding algorithm has been thoroughly investigated and has been subject to much experimentation.

Forwarding of requests take place so that the requests propagate towards more "authoritative" nodes and eventually end up in a node which knows "everything", i.e. the node which we denote as the *root VA*. To accomplish this, we choose to organize the overlay network as a *shortest path spanning tree*[3] (SPST). The root VA (the MANET node with knowledge about every certificate) becomes the root of the SPST, and the tree is constructed so that every node has the shortest path possible to the root. The cost of each link in the tree is calculated as the number of hops between the two nodes in the underlying MANET.

Due to the dynamic nature of the MANET topology, the SPST must be calculated for each forwarding operation. The chosen node for forwarding requests is the parent node in the tree. Consequently, the request travels "upwards" in the tree towards the root VA, which is the ultimate target of its migration and the node which is able to process every validation request. The result message (POV) from the validation, whatever node on the path is able to process the request, travel downwards in the tree towards the originator, and is cached by every proxy node it passes through.

As earlier pointed out, only the root VA issues POVs. The other nodes merely caches and transports them. The clients' trust lies with the digital signature of the root VA, no one else in the proxy network is trusted.

[3]A shortest path spanning tree (SPST) is different from a minimum spanning tree (MST). While an MST minimizes the total weight of all the links in the tree, an SPST will minimize the path length from the root to any node.

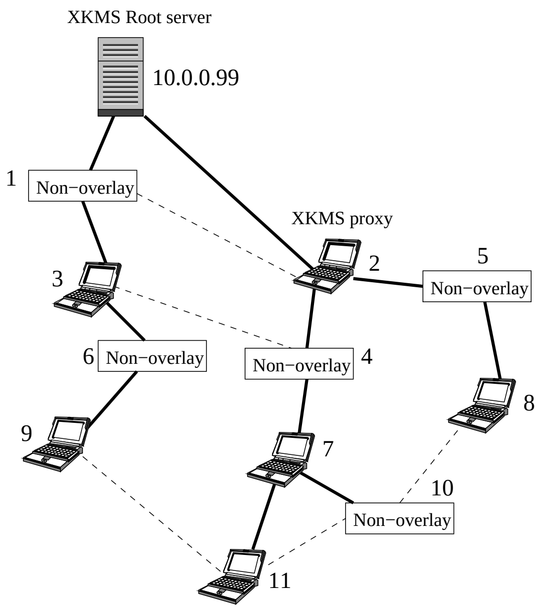

Figure 2.1: XKMS overlay proxy network, formed as a Shortest Path Spanning Tree (SPST) The numbers refer to the IP addresses of the nodes

2.2.1 Disconnected operation

The algorithm just described works fine when the MANET topology graph is connected, i.e. every node has a path to the root VA node. All requests are eventually processed, as they would have been if there were no proxy nodes present, and all requests were passed to the the root VA by the MANET routing layer. The effect of the proxy nodes in a connected network is reduced network traffic, in particular on the links close to the root VA.

A MANET is often *partitioned*, however, meaning that the network topology forms isolated islands and that nodes have connection paths to only a subset of the MANET nodes. For those nodes who do not have a path to the root VA, a proxy node within reach may offer a successful validation operation. It is one of the hypotheses of this report that the presence of proxy nodes will increase the availability of the validation service and increase the success rate of clients' validation requests. The hypothesis has been investigated by experiments which will be presented later in the report.

To *whom* should we forward the request? A "partitioned node" (the term referring to a node without a route to the root VA) is not able to construct a SPST. The forwarding decision must therefore be an educated guess on the basis of present topology or past forwarding decisions. During the experimental evaluation of the proxy overlay network two approaches have been investigated:

1. Forward to the same node as last time (the "last known parent")

2. Choose the closest neighbor (smallest number of hops away)

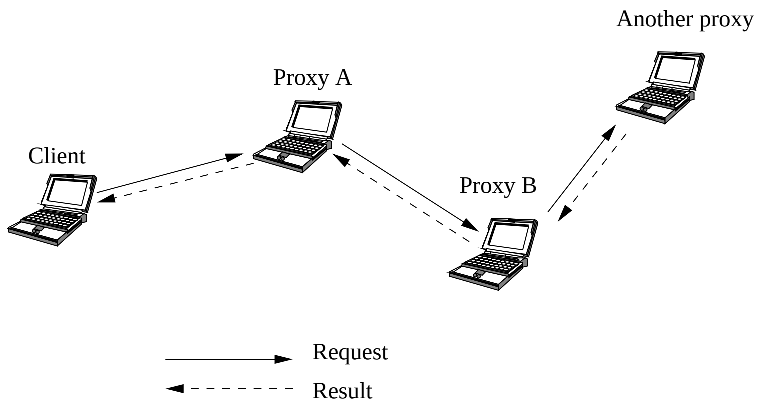

Figure 2.2: Forwarding of validation request (solid lines). During each step in the forwarding path, the node forwarding the request becomes client of the addressed node.

While forward decisions based on a SPST guarantee a loop-free forwarding path, the heuristic decisions made by a partitioned node *will* generate loops; Approach no. 2 from the list above used in e.g. a network partition with two nodes will cause the message to bounce endlessly between them. When these heuristics were implemented in the prototype, careful coding was necessary to detect and discard looping messages. There were some interesting surprises related to these efforts which will be described in the experimental section of the report.

2.3 Necessary changes to existing protocols

The XKMS standard was chosen as the protocol for validation operations. XKMS is based on the use of XML for encoding of message content and SOAP protocol for message transport. HTTP was chosen as the application protocol for transporting SOAP messages. The XKMS specifications had to be slightly modified for the use in a proxy overlay network. Some knowledge about the XKMS specification[8] is required for the discussion to follow:

During a forwarding operation, the message is passed on unaltered. This means that one proxy node (or the root VA) will receive a message which was originally addressed to a different node, and the result message (POV) is generated by a different node than what was addressed.

Since the caches are supposed to hold recent results only, the root VA must add an expiration time to the POV at issuing time[4]. Timestamps are not included as a part of "standard" XKMS messages. Necessary changes and amendments are:

- The XKMS server (root VA or proxy node) must accept `ValidateRequest` messages with a `Service` attribute value different from the URL of its own service endpoint.

[4]An issuing timestamp on the POV would allow the client to set its own recency requirements, but that does not make sense in an environment of cooperating caching.

- The XKMS client must accept a POV in the form of a `ValidateResult` message with a `RequestId` attribute value different from the `Id` attribute value in the originating `ValidateRequest` message.

- The `ValidateResult` message type must employ the optional `MessageExtension` element to include an expiration timestamp.

- Since the transport of POVs takes place during non-authenticated sessions, the `ValidateResult` message must be signed by the root VA, i.e. the `Signature` element is mandatory.

These adaptations to the protocol are not likely to be offered by any existing XKMS server. It was therefore necessary to write an XKMS server as a part of this experiment. This server runs as the root VA.

2.4 The use of a cross layer design

The proxy node needs to build the SPST as a part of its forward decision, and to know the distance to any other node. In many peer-to-peer (p2p) distribution projects this information is gathered through a *peer discovery protocol*, which require regular "heartbeat" messages to travel the network and consume bandwidth. Peer discovery protocols duplicate the functions performed by a link state routing protocol, which also need to sense changes in link states and flood the network with updated information. The forward decisions would not need a separate discovery protocol if the proxy node could access the information stored in the routing protocol software.

The term *cross layer design* refers to connections between software modules outside the traditional service interface. In the context of a communication protocol stack, cross layer means that one protocol layer offers access to its internal sensors and data structures to a different layer.

The implementation of the proxy node needs access to the link information in order to build a SPST. In addition, it needs to distinguish ordinary MANET nodes from proxy nodes, since only the proxy nodes are members of the SPST. To accomplish this, a cross layer design was employed through the modification of an existing MANET routing protocol and an existing routing executable.

For the MANET routing the *Optimized Link State Routing* (OLSR) protocol was chosen. This protocol has been developed with MANET operation in mind, and offers an economic and efficient broadcast algorithm for distribution of link state information etc.

There exists an open source implementation of OLSR[5] available as source code which can be modified for the purpose at hand. The modification of the protocol and executable code is presented in [5], and this report provides only a summary:

[5]Available from `http://www.olsr.org/`

- Every MANET node is allowed to set a 16 bit variable called `OverlayMembership` which is distributed to every other MANET node through a modified HELO and TC (Topology Control) OLSR message. The value of this variable is given in the OLSR configuration file. In the present experiment, bit 0 of the variable expresses that the node is a member of the proxy overlay network, and bit 1 indicates that this node is a root VA (and becomes the root of the SPST).

- The modified OLSR executive offers an XML representation of its internal data structures through a web services interface. A local program (e.g. the proxy node program) can connect to it with HTTP protocol and fetch the state of the links in the network, the resulting routing table, the local IP address and netmask etc. Also, it can retrieve the value of the `OverlayMembership` word in each node.

The price to pay for cross layer design is tighter coupling between software components. The present proxy node implementation does not work unless this particular OLSR executive is in operation, and all nodes in the MANET must run the same executive since the routing message layout has been altered.

Figure 2.3 shows how the OLSR offers its data to a web browser. The name of the elements and attributes should be self explanatory.

3 Implementation of design prototype

The design described in the previous chapter has been implemented for the purpose of experimental evaluation.

3.1 XKMS server

The XKMS server used as a root VA has been implemented as a Java Servlet and deployed in a Jetty web server[6], although any servlet container could have been used. The XKMS server implements the KISS protocols only (Key Information Service Specification) and can be used to locate and validate certificates and keys. Its “back end” accesses a PKI through a LDAP interface. For each validation operation, the server fetches the entire certificate path (up to the trust anchor) and checks signatures, usage attributes, validity intervals etc. Since every validation is based on on line information from the PKI directory service, the use of status providers (OCSP) or revocation lists is deemed unnecessary.

[6]Available from `http://www.mortbay.org/`

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.ffi.no/olsr">
<m:config>
 <m:mainaddress>10.0.0.07</m:mainaddress>
 <m:ipv4conf name="eth0">
  <m:addr>10.0.0.07</m:addr>
  <m:mask>255.255.255.0</m:mask>
  <m:bcast>10.0.0.255</m:bcast>
 </m:ipv4conf>
</m:config>
<m:topology>
 <m:edge dest="10.0.0.99" lastHop="10.0.0.01" destOverlay="3"/>
 <m:edge dest="10.0.0.99" lastHop="10.0.0.02" destOverlay="3"/>

 <m:edge dest="10.0.0.01" lastHop="10.0.0.99" destOverlay="0"/>
 <m:edge dest="10.0.0.01" lastHop="10.0.0.02" destOverlay="0"/>
 <m:edge dest="10.0.0.01" lastHop="10.0.0.03" destOverlay="0"/>

 <m:edge dest="10.0.0.02" lastHop="10.0.0.99" destOverlay="1"/>
 <m:edge dest="10.0.0.02" lastHop="10.0.0.01" destOverlay="1"/>
 <m:edge dest="10.0.0.02" lastHop="10.0.0.04" destOverlay="1"/>
 <m:edge dest="10.0.0.02" lastHop="10.0.0.05" destOverlay="1"/>

 <m:edge dest="10.0.0.03" lastHop="10.0.0.01" destOverlay="1"/>
 <m:edge dest="10.0.0.03" lastHop="10.0.0.04" destOverlay="1"/>
 <m:edge dest="10.0.0.03" lastHop="10.0.0.06" destOverlay="1"/>

 <m:edge dest="10.0.0.04" lastHop="10.0.0.02" destOverlay="0"/>
 <m:edge dest="10.0.0.04" lastHop="10.0.0.03" destOverlay="0"/>
 <m:edge dest="10.0.0.04" lastHop="10.0.0.07" destOverlay="0"/>

 <m:edge dest="10.0.0.05" lastHop="10.0.0.02" destOverlay="0"/>
 <m:edge dest="10.0.0.05" lastHop="10.0.0.08" destOverlay="0"/>

 <m:edge dest="10.0.0.06" lastHop="10.0.0.03" destOverlay="0"/>
 <m:edge dest="10.0.0.06" lastHop="10.0.0.07" destOverlay="0"/>
 <m:edge dest="10.0.0.06" lastHop="10.0.0.09" destOverlay="0"/>

 <m:edge dest="10.0.0.07" lastHop="10.0.0.04" destOverlay="1"/>
 <m:edge dest="10.0.0.07" lastHop="10.0.0.06" destOverlay="1"/>
 <m:edge dest="10.0.0.07" lastHop="10.0.0.10" destOverlay="1"/>
 <m:edge dest="10.0.0.07" lastHop="10.0.0.11" destOverlay="1"/>

 <m:edge dest="10.0.0.08" lastHop="10.0.0.05" destOverlay="1"/>
 <m:edge dest="10.0.0.08" lastHop="10.0.0.10" destOverlay="1"/>

 <m:edge dest="10.0.0.09" lastHop="10.0.0.06" destOverlay="1"/>
 <m:edge dest="10.0.0.09" lastHop="10.0.0.11" destOverlay="1"/>

 <m:edge dest="10.0.0.10" lastHop="10.0.0.07" destOverlay="0"/>
 <m:edge dest="10.0.0.10" lastHop="10.0.0.08" destOverlay="0"/>
 <m:edge dest="10.0.0.10" lastHop="10.0.0.11" destOverlay="0"/>

 <m:edge dest="10.0.0.11" lastHop="10.0.0.07" destOverlay="1"/>
 <m:edge dest="10.0.0.11" lastHop="10.0.0.09" destOverlay="1"/>
 <m:edge dest="10.0.0.11" lastHop="10.0.0.10" destOverlay="1"/>
</m:topology>
</soap:Body>
</soap:Envelope>
```

Figure 2.3: SOAP message returned from the OLSR executive describing the network shown on Figure 2.1

3.2 XKMS proxy node

The XKMS proxy node implements the XKMS interface and is also programmed in Java. It is not implemented as a Java servlet, for reasons of resource efficiency. The proxy node is programmed with its own embedded HTTP server so that it can be run directly from the command line. This approach reduces the software installation footprint and offers better control over the TCP socket operations.

The proxy node is dependent on the presence of the modified OLSR executive from which it retrieves topology information during forwarding operations. The proxy node does very little processing of request and result messages, and merely inspects them for its processing decisions. During forwarding the messages are not changed in any manner, but are passed on literally.

3.3 OLSR executive

The OLSR executive is written in C and has been modified for the purpose of proxy node discovery and external access to topology information. The changes are documented in the source code and should be easily repeated on more recent versions of the software.

3.4 Portability considerations

The Java code in the XKMS processors can be run on a wide range of platforms, including Windows and Linux. The OLSR executive has been compiled for Linux and Windows (using the Cygwin C compiler, not Visual Studio). The overlay proxy network can therefore run on a mixed platform of Linux and Windows computers.

3.5 Intellectual property rights

The source code bears no copyrights, trade secrets or classified information, and can be freely used in any project.

4 Evaluation of the design prototype

The design described in the previous section has been implemented in an experimental prototype and evaluated for its efficiency. This section will provide a presentation of the experimental design and the result of the experiment.

4.1 Chosen values for observation

The design of the XKMS proxy network was expected to have two effects:

- Reduction of network traffic, in particular in the region close to the root VA
- Improved availability under mobile conditions, i.e. more validation requests are successful

In order to assess these expectations, the chosen variables for observations were:

1. The number of requests received by the root VA
2. The fraction of client requests leading to a successful validation operation

In addition to these variables, some variables were observed in order to investigate the inner mechanisms of the proxy algorithm:

3. The counter of the event where a forwarding operation successfully validated a certificate in "partitioned mode" through the "last known parent" of the requesting node (cf. Section 2.2.1)
4. Same as above, but forwarded through the "nearest neighbor"
5. The number of forwarded operations in "partitioned mode" that was unsuccessful (the forwarding operation was successful, but the validation operation did not take place)
6. The number of connect or read timeout on the TCP socket operations

4.2 Controlled variables

The observed variables were expected to be dependent on a number of controlled variables which were given different values during the experiment.

The first controlled variable in this experiment was the presence of proxy nodes. Without any proxy nodes each client addresses its requests directly to the root VA, and the requests are routed through the MANET by the underlying routing layer. This will resemble an "ordinary" validation configuration (with no intermediaries between the client and the VA), and this configuration was chosen as the baseline of the evaluation. With proxy nodes present, the same evaluation scenario is played again and the dependent variables observed.

Figure 4.1 shows the two configurations: At (a), there is no proxy overlay, and all client request are sent directly to the root VA. With a proxy node configured, as shown in (b) the request are forwarded to the parent node in the SPST and migrates towards the root VA.

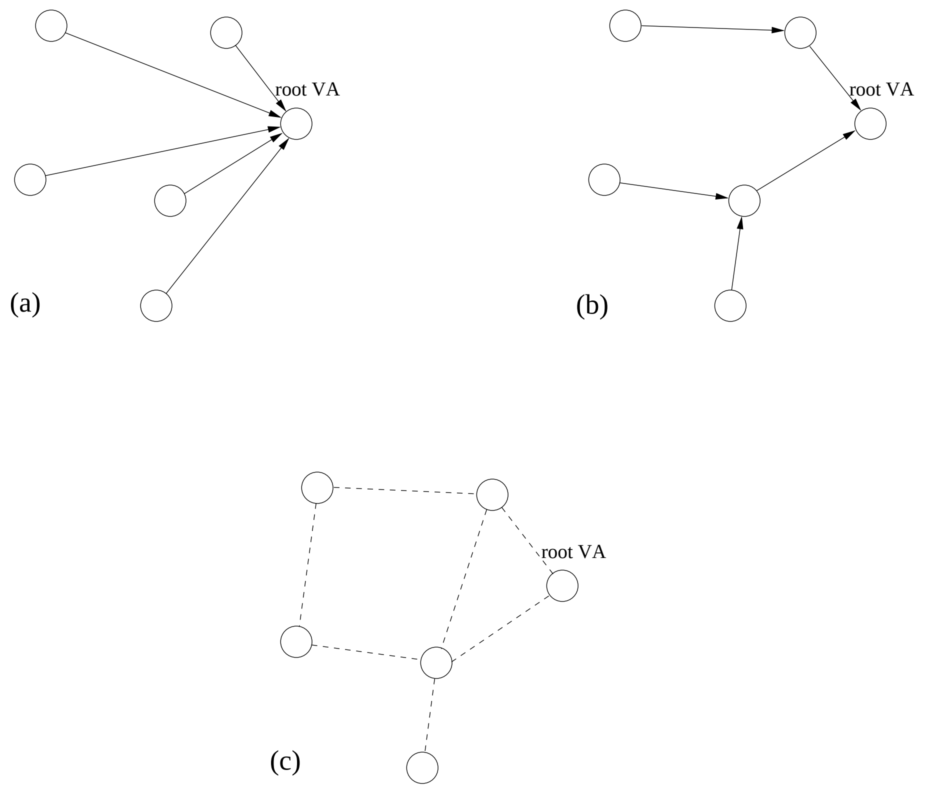

Figure 4.1: Without a proxy overlay, all requests are sent directly to the root VA (a). With a proxy overlay present (b), the requests are sent stepwise towards the root VA. The graph at (c) indicate a possible link topology which would result in the SPST shown in (b)

Please note that this relates to message forwarding on application level, and does not indicate any particular link topology. Part (c) of Figure 4.1 shows a possible link topology that will form an SPST shown in part (b).

The second controlled variable is the mobility model and mobility parameters. Different mobility scenarios were applied to similar validation experiments and the results compared.

4.3 Invariants

The remaining properties of the experimental environment has been kept invariant to the extent possible.

The term “to the extent possible” is used since the experiment contains a number of stochastic and asynchronous processes, and the order of which they start and finish affects the final result of the experiment. The sources of errors and inaccuracies which have been identified are discussed in Section 4.10.

Table 4.1 lists some key parameters and their values during the experiment.

Parameter	**Value**
Number of MANET nodes	24
Number of proxy nodes	23
Size of certificate population	100
Number of validation requests per node	100
Mobility recalculation interval	2000 ms
Mobility random number seed	5
TCP connect timeout	2000 ms
TCP read timeout	5000 ms

Table 4.1: Constant values of experiment parameters

4.4 Mobility scenarios

The mobility of the MANET nodes was emulated by a *mobility model* which is a part of the emulation testbed described in Section 4.6. The mobility scenarios were set up to be identical for each experimental run. The chosen scenarios were rather “demanding” with frequent disconnected links and partitioning of the network.

The mobility scenarios used an area 640x480 pixels. Distances were calculated with the side length of a pixel as the length unit.

Three different mobility models were employed, all with 24 participating nodes:

1. A hierarchical configuration consisting of:
    - one stationary command station hosting the root VA service
    - two aircrafts, roaming around in the entire simulation area using the “random waypoint” mobility model [2]
    - three ground vehicles, moving around the entire simulation area using the “random waypoint” model (but much slower than the aircrafts)
    - six soldiers per vehicle, roaming around in a area of 50x50 pixels centered around the vehicle
2. A hierarchical configuration similar to the one above, with the following differences:
    - soldiers roam an area of 30x30 pixels
    - the VHF radios have a range of 300 pixels
3. A flat organization of 23 soldiers roaming around a command station with successively larger area (from 10x10 to 150x150 pixels). The command post is stationary, and the soldiers move according to a “random waypoint” mobility model

The nodes were equipped with the following radios:

Node	**Radio**
Soldier	UHF
Vehicle	VHF,UHF
Command station	VHF
Aircraft	VHF

Table 4.2: Radio equipment in use

The radios had the following ranges:

Radio	**Range (pixels)**
UHF	20
VHF	200 or 300

Table 4.3: Range of radio equipment

4.5 Validation scenario

It was also necessary to decide the pattern for validation requests. It is improbable that every certificate is validated with the same frequency. From analysis of different social networks [1] a *scale free distribution* has been proposed to describe the frequency of social interactions. In scale free distribution objects are *ranked*, and the frequency[7] of each object is estimated as a function of its rank according to the following equation:

$$freq(obj) = \frac{a}{rank(obj)} \tag{4.1}$$

Where a is given a value so that

$$\sum_{obj} freq(obj) = 1 \tag{4.2}$$

Since the validation of certificates is related to message transactions, we have chosen to rank all certificates and assign them a validation frequency according to Equation 4.1. Certificate r will thus be validated twice as often as certificate $2r$. All clients rank to the individual certificates in the same order, meaning that the relative frequency of certificate validations will be the same in all client nodes. The actual choice of certificates for validation is a random process (according to the probability distribution in Equation 4.1), so the order of validation operations is unknown in advance.

4.6 The MANET emulation testbed

The evaluation of the XKMS proxy overlay was done in a MANET emulation testbed which was constructed for the purpose of this experiment. The testbed consisted of 24 virtual computers hosted on 6 physical computers. The computers were connected through a wired LAN. An extra "conductor" node was running the mobility model described in Section 4.4, and was broadcasting the connectivity matrix of the model at regular interval (the mobility recalculation interval from Table 4.1). In each node[8] an agent program interpreted the matrix and set up MAC filters on who to accept Ethernet frames from. This report only gives this brief description, since the MANET testbed is presented in a separate FFI memo [6]. This memo also provides details on how the experiment was conducted, how the programs were started and how the values of the observed variables were gathered.

4.7 Observations during the experiments

Each node reported 6 different events to the central node:

1. Failed validation operation
2. Successful validation
3. Successful validation through "Nearest neighbor"

[7]e.g. frequency of social interactions like a phone call or an e-mail message
[8]The term "node" from now on refers to a virtual node.

4. Successful validation through “Last Known Parent”
5. Request was forwarded in partitioned mode (to either nearest neighbor or last known parent), but this neighbor was not able to provide a validation result
6. Socket timeout during forwarding operation

The request messages reaching the root VA were also counted and shown on the console of the root VA. This number is an important indicator on the reduction in network traffic caused by the caching algorithm in the proxy nodes.

A discussion on the interpretation of these events are now in order: Events 1 and 2 are client-centric events (issued by the originating client), in the sense that they report end results from the XKMS proxy service. An unsuccessful validation is reported (Event 1) if the client receives a SOAP Fault reply, or a socket timeout/exception is thrown.

Events 3 and 4 relate to validation operations in “partitioned mode”, i.e. when the client does not have a route to the VA and consequently cannot build a SPST. In these situations, the client forwards to the node which was its last SPST parent if there exists a route to it. It not, the client picks the closest node in the network partition (fewest hops away). When these forwarding operations result in a successful validation, they are reported as event 3 or 4. Please observe that the same validation operation may cause this event to be reported from more than one node along the forwarding path.

Event 5 is reported if the forwarding operation during “partitioned mode” was successful (meaning that the target node processed the request), but the result was unsuccessful (the response was a SOAP Fault message). This event may also be generated by several nodes along the forwarding path for the same validation operation.

Event 6 is reported by a node who attempts to set up a TCP connection to another node, resulting in a `java.net.SocketTimeoutException`. The timeout values are shown in Table 4.1.

Observe that event 1/2 are mutually exclusive, as well as 3/4 and 2/5. Observe also that the same validation operation may cause several events, in the same or in different nodes. Event 1 would likely occur as a result of event 6 in a different node. Thus, the events are not interpreted as a “state” variable.

In the log file of the experiment, the events are presented with the IP address of the sender. No information about the identity of the request messages are stored.

These events are reported in order to study these questions:

- How often were validation operations successfully obtained in “partitioned mode”?
- How often is the “closest neighbor” able to provide a validation service?
- How often are operations terminated by a socket timeout?

The answer to these questions will provide some insight in the “inner workings” of the proxy network, and provide a basis for experimentation on different design alternatives and different mobility models.

4.8 The endless loop phenomenon

The choice of “nearest neighbor” will inevitably lead to loops in the forwarding of requests (Section 2.2.1). Therefore, the proxy program keep a small memory (30 entries) if previously forwarded messages and will discard messages who turn up the second time. During experimentation, loops did occur despite this mechanism, and analysis of the network traffic revealed a rather large row of messages (around 100) that was looping between two nodes. Besides, the cycle period for one message (approx. 1 minute) was by far exceeding the timeout mechanisms employed in the forwarding operation. This phenomenon was highly unexpected and had to be investigated.

One key observation during the investigation is that the application cannot control the number of incoming TCP connections: Even if the application does not listen for a connection (using the `ServerSocket.listen()` call), the operating system allows a number of incoming calls the be processed with the three-way TCP connection handshake, and will also allow incoming messages to be sent, acknowledged and buffered. The application does not notice that the incoming calls have been buffered in this way, and the caller will not block its operation until it attempts to read the response from the application. The size of the queue for incoming calls varies between operating systems, and have been observed to be in the range between 20 and 50 incoming calls.

The short form of the explanation is that this queue is longer than the size of the duplicate memory used to prevent looping. The longer form is presented on Figure 4.2 which will be explained in the following paragraphs.

The figure has a vertical time scale and shows the messages between two proxies *a* and *b*. They belong to the same partition, and has each other as closest neighbor, and will therefor attempt to pass messages back and forth between themselves.

The sequence of events are as follows:

1. *a* send request *Req* to *b*

2. *b* processes the message (checks its own cache) and forwards it to *a*. *Req* and *Req*′ has the same content. Now *b* waits for the response from *a*, blocking its thread in a read operation with a timeout clause.

3. *b* has no available server thread (it does not listen to connections), therefore the operating system accepts the connection and buffers the message on behalf of the proxy.

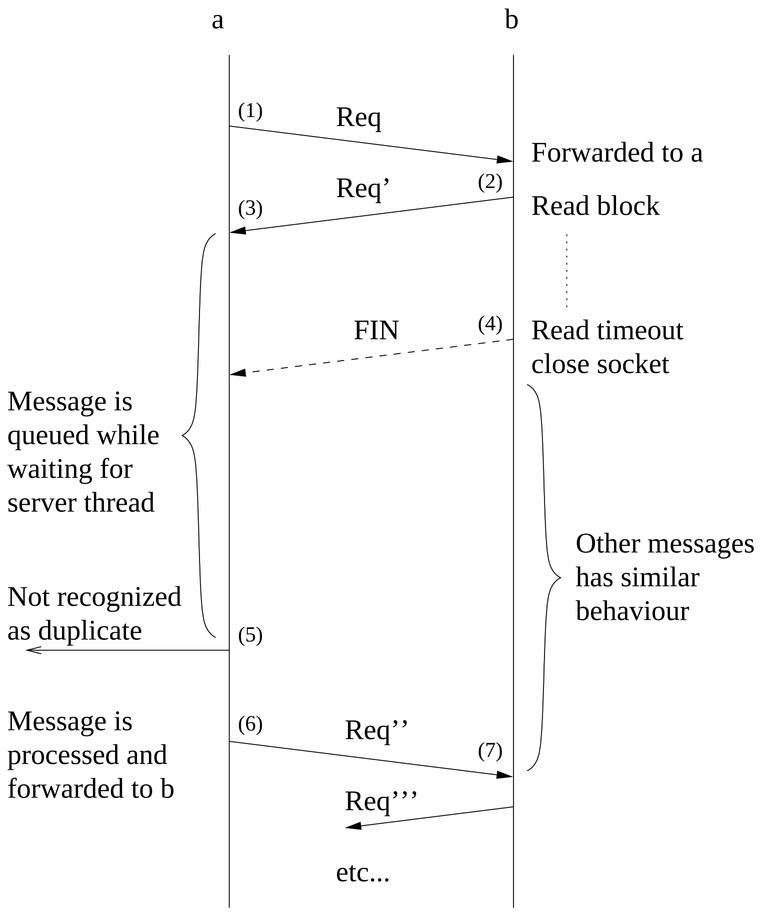

Figure 4.2: The sequence of events leading to the endless loop. The numbers refers to the explanation in the text

4. The queue at *a* is long and the delay is considerable, so in the mean time, the read operation times out and throws an exception in the proxy program. The result is that the socket is closed and the proxy program returns a soap fault to its caller. The FIN message shown is a part of the TCP protocol's close operation.

5. As *a* later listens for a connection, it will receive the incoming connection and the buffered message even if *b* has closed its socket. This semantics is acceptable in a general context, but in the proxy overlay operation, where the response is a mandatory part of the protocol, it does not make sense to process a message received under these conditions.

6. In the period between (3) and (5), *a* receives several incoming messages, so the duplicate detection memory is overrun. Therefore the message is processed as a new message and forwarded to *b*.

7. The relation between *a* and *b* is symmetric, so the same situation occurs at *b*: The duplicate detection memory is overrun so the message is processed as a new message.

The fault message generated from the timeout occurring at (4) is never received, since the sender has closed its socket on a a previous timeout occasion.

The cure for this phenomenon is:

- To ensure that the socket is connected when a message is received. The `Socket` class offers a `isConnected()` method which will indicate if the counterpart has closed its socket, and the receiver should discard the message under such conditions.

- To increase the size of the duplicate detection memory (to 100 entries).

4.9 The experimental runs and results

The experiment runs were conducted as follows:

1. The mobility model was started

2. The test program on each of the 23 overlay nodes was instructed to:

   (a) pick a certificate from a population of 100 certificates according to a scale-free distribution

   (b) pass this certificate to its service point for validation with the XKMS protocol.

   (c) Log the result (event 1 or 2)

   and to repeat this (inner) list of steps 100 times.

Repeat this (outer) list a number of times and calculate the average and standard deviation of the event counters described in Section 4.7.

Experimental runs were executed with the three different mobility scenarios shown in Section 4.4, and with two XKMS validation service points: The local XKMS server (the proxy residing on `localhost`) and the XKMS server running on the root VA. These two service points represent the validation operations with or without the proxy overlay, respectively.

The results from 4 of these 6 sets of experimental runs are presented in Table 4.4 (the other two sets are given different conditions and is discussed later in the section). The columns show the observed event counters by their average values and standard deviation. The rightmost column is the number of request messages received by the root VA. The different rows shows the result for mobility scenario no. 1 and 2, respectively.

Event no.	ev 1	ev 2	ev 3	ev 4	ev 5	ev 6	#root VA
Mobility scenario 1:							
With proxy:							
Avg (n=17)	888	1370	40	284	572	732	262
Std dev	95	75	69	75	192	97	32
Without proxies:							
Avg (n=8)	1080	1209					1243
Std dev	60	51					52
Mobility scenario 2:							
With proxy:							
Avg (n=11)	320	1980	1	213	64	399	386
Std dev	19	19	3	143	31	23	9
Without proxies:							
Avg (n=11)	476	1824					1901
Std dev	22	22					24

Table 4.4: Table of experimental results

The counter values for event no. 2 is shown in Figure 4.3, and the counter values for request to the root VA server is shown in Figure 4.4.

The most important columns are showing event 1 and 2. Their relation show the service level improvement caused by the proxy overlay: For mobility scenario no. 1, the improvement is:

$$\frac{1370/(1370+888)}{1209/(1209+1080)} = 1.181 \tag{4.3}$$

For mobility scenario no. 2, the improvement is:

$$\frac{1980/(1980+320)}{1824/(1824+476)} = 1.084 \tag{4.4}$$

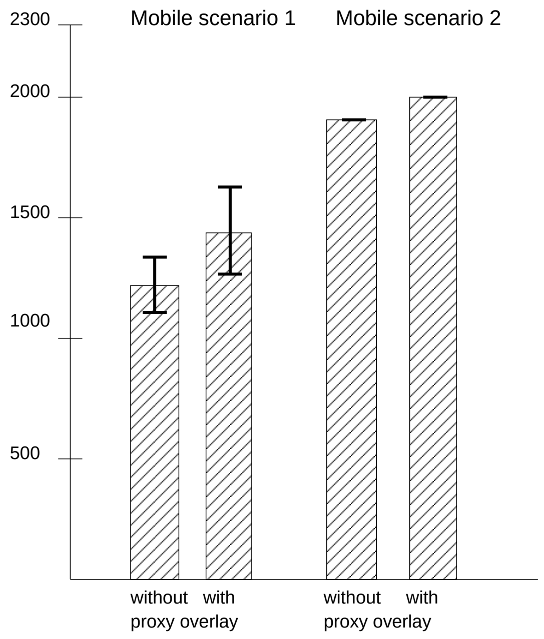

Figure 4.3: A graphical representation of the diffence in service availability caused by the presence of the proxy overlay. The 0.95 confidence interval are shown at the top of the bars, and indicates that the observed differences for mobility scenario 1 is not significant

It may seem that mobility scenario no. 1 shows the best improvement, but due to the larger standard deviation, this result is not statistically significant.

The values of the columns for event no. 3, 4 and 5 indicate the conditions when the network is partitioned. Since this condition only applies to the proxy overlay mechanism, the values are not listed for non-overlay operation. The values for event no. 5 are consistent with the values of event no. 1 (given the large standard deviation), and they are related since they both apply to unsuccessful validation operations. The columns indicate that the number of operations where an “nearest neighbor” offers a successful operation are few (event 3), related to the number of operations successfully solved by the “last known neighbor” (event 4). The rightmost column, showing the number of messages received by the root VA, shows a clear and large reduction of network traffic to the root VA due to the proxy overlay.

The experiment runs with mobility scenario no. 3 were given slightly different conditions: Only one experimental run (as described in Section 4.9) were executed for the two possible proxy configurations (with and without proxy overlay). Between each run, the mobility parameters were changed so that the 23 soldiers were given a successively larger area for their movements. As expected, the performance of the validation service decreases as this area grows, due to more frequent link disconnections (radios out of range due to larger distances). The result from mobility scenario no. 3 can be seen on Figure 4.5. The vertical axis represents the number of successful validation operations (maximum 2300). The horizontal axis represents the area in which the 23 mobile nodes were allowed to roam, ranging from 10x10 to 150x150 pixels. The two lines indicate the performance

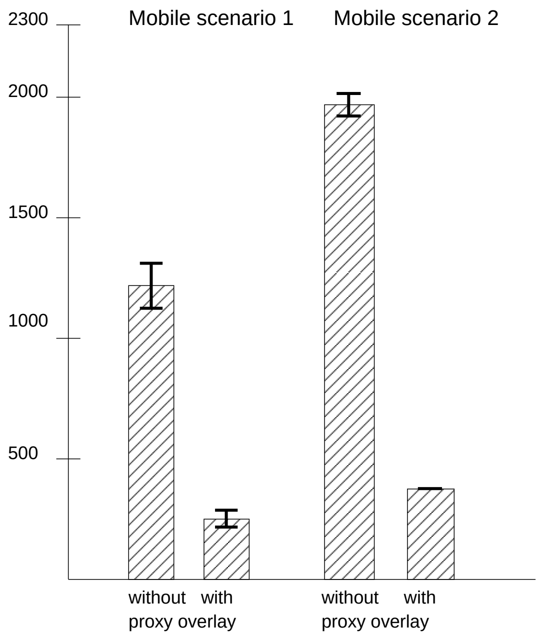

Figure 4.4: A graphical representation of the diffence in network traffic to the root VA server. The 0.95 confidence interval are shown at the top of the bars. The reduction in traffic is large and significant for both mobility scenarios.

with or without the presence of a proxy overlay.

The graph shows that the performance is consistently better with the proxy overlay present as long as the roaming area is small. As the roaming area grows larger, there are several observations where the proxy overlay results in a poorer performance. The increasing raggedness of the line indicates also that the variation of the observations is becoming larger, which is consistent with the values in Table 4.4. We cannot therefore conclude that the result of the proxy overlay is negative under these condition, but we cannot offer any other explanation.

4.10 Error sources

There are several factors that introduce errors and inaccuracies which have been identified during the experiment. They will be presented in this section:

- The sequence of validation requests, issued by the 23 clients, is not uniformly distributed in time. An unsuccessful validation operation ending in a "Socket Timeout" will take 2 seconds (the timeout interval) to complete, while a successful operation may take much shorter. Consequently, there tends to be a higher rate of validation operations during connected periods (where successful validation operations can take place) than during partitioned periods. The results may therefore a show a higher success rate than if the request were strictly isochronous.

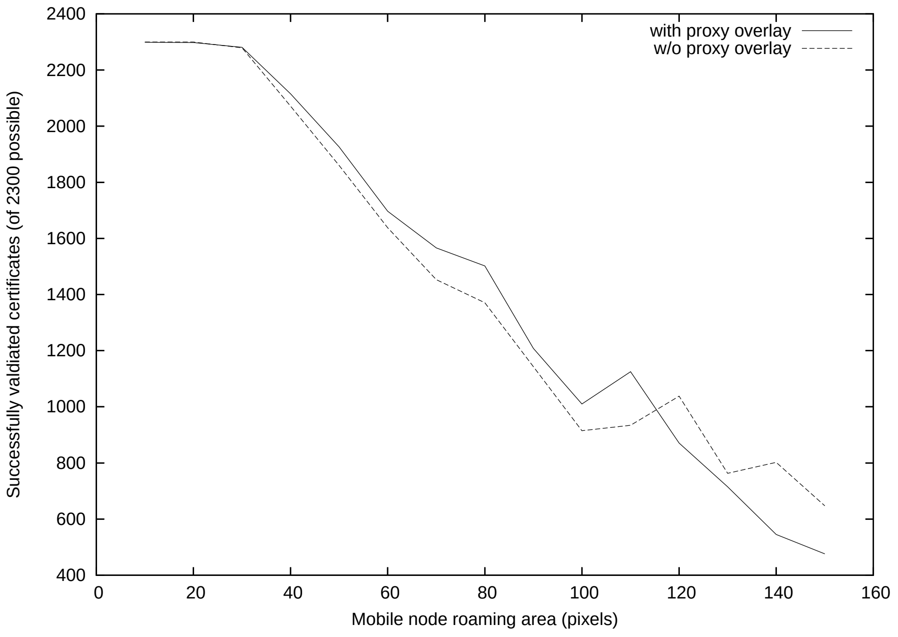

Figure 4.5: Performance of the proxy overlay under mobility scenario no. 3

- The proxy overlay node program sometimes stopped without any apparent reason, so the total number of requests (sum of events 1 and 2) is less than 2300.
- The relatively short socket timeout limits (2 seconds for connect, 5 seconds for read) may have caused queuing delays to be erroneously diagnosed as connectivity problems. The proxy nodes offer their XKMS interface on a server with a thread pool for connection handling. If several connections are made at the same time, some queuing delay will occur.
- Clock skew affects the lifetime of the cached POVs, since their timestamps are issued by the root VA, but checked by the proxy nodes. Although the clock values were regularly aligned, we must assume that clock skew has reduced the accuracy of the observations.
- The emulation testbed presented in Section 4.6 does not limit the bit rate. All communication happens at Ethernet speed (100 Mbps) which is unrealistic for a military MANET. As a compensation, the speed of mobility is set higher than what is realistic.
- The topology table fetched from the OLSR executive were sometimes in a transitional and inconsistent state. In these situations, the proxy node could not compute a SPST, and concluded that the node was in partitioned mode. The OLSR executive does not offer any transactional isolation of its operation on the internal data structures, so this situation seems unavoidable, and have probably caused the proxy node to operate in partitioned mode unnecessarily.

These error sources affect all observations. Since the experiment takes a comparative approach, the effects may have been canceled to some extent. The final effect of these factors still remain unknown, though.

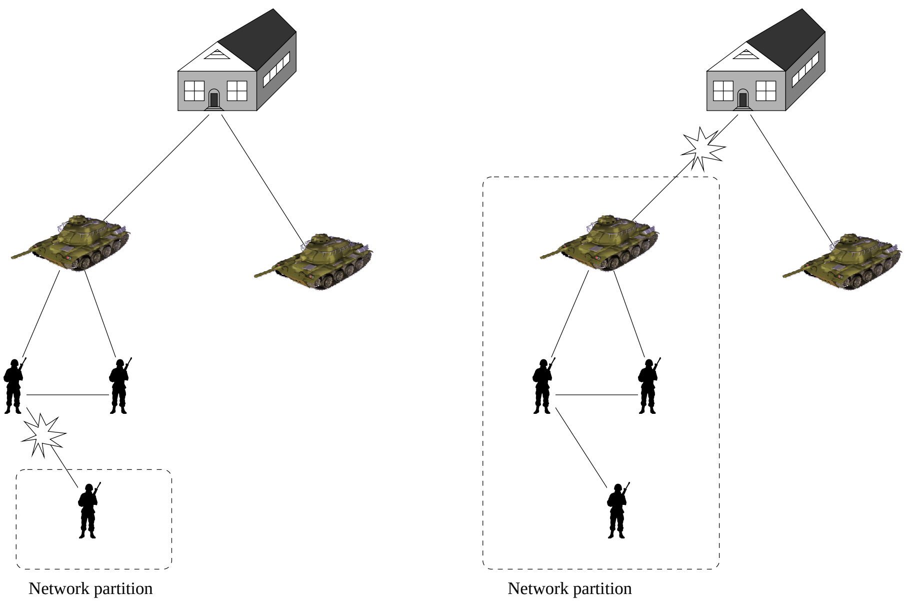

Figure 4.6: Broken links low in the mobility hierarchy (left) creates a smaller network partition than broken links higher up (right)

4.11 Discussion of the observations

The observations presented in Section 4.9 were made in an emulated environment, meaning that several factors which would impact a "real world" system is not measured. The purpose of the experiment, however, is to study the effect of a *dynamic topology*, not radio channel performance.

The effect of a proxy overlay was expected to be (1) reduced network traffic and (2) improved service availability. The effect on network traffic has been observed to be significant and large. The observed effect on service availability has been small, and depends on the mobility scenario.

From an analytical perspective, the expected effect of a proxy overlay in a network partition will be larger if the network partition contains several proxy nodes. A larger partition will contain more cached POVs and offer a higher probability of a successful validation based on a cached POV, provided that the caches can be be exploited by the client. We will therefore inspect the mobility scenarios to see if they are different in this respect

In a hierarchical mobility model, as used in mobility scenario 1 and 2 (Section 4.4) soldiers are clustered around their respective vehicles, and vehicles clustered around the command post. It is a reasonable assumption that broken links high up in the hierarchy creates a larger network partition disconnected from the root than a broken link far down in the hierarchy. As a consequence, a proxy network where most lost links occur at the vehicle-commandpost level are likely to offer better availability of validation services than if links are broken mostly on the soldier-soldier and soldier-vehicle level. The two situations are illustrated in Figure 4.6.

Mobility scenarios 1 and 2 are somewhat different in this respect: In mobility scenario 1 the relation between the allowed roaming distances and the range of their radio equipment (See Section 4.4 for values) suggests that broken links are more likely to occur on soldiers-soldiers link and soldiers-vehicle link than on the vehicle-commandpost link. For scenario 2 the range of VHF links has been deemed shorter and the roaming area for soldiers has been made smaller for the purpose of shifting the probability for broken links to the VHF links (the vehicle-commandpost links). The assumption made is that scenario 2 will generate larger network partitions, and the observed results supports that assumption.

5 Conclusion

This report has investigated the effects of a proxy overlay on a distributed certificate validation service.

The design of the proxy overlay was expected to have positive effects on a certificate validation service, both in terms of reduced network traffic and improved availability. The conclusions of the experimental evaluation of the design are:

- The introduction of a proxy overlay causes a large reduction in the network traffic necessary to offer a validation service
- The proxy overlay also causes an improved availability of a validation service, but with a smaller margin. The improvement is also quite dependent upon the mobility scenario.

It is only when a node is disconnected from the root VA that the proxy overlay has effect on the availability of the validation service. The effect has further been observed to depend on the number of other reachable nodes in the partition; A higher number of nodes in the network partition increases the probability of a successful validation operation.

The design of a cooperative caching mechanism with on-demand replication of validations results have relevance for other client-server based applications, where similar requests get similar responses if the recency requirements are met. The results from the study indicate what kind of performance effect a similar arrangement may have in a mobile context, e.g. a distributed service discovery system.

6 Acronyms and Abbreviations

HTTP HyperText Transport Protocol, the application protocol used for transport in Web applications.

LDAP Lightweight Directory Access Protocol, the application protocol used for access to directory systems.

MANET Mobile Ad Hoc NETwork, a term used for a communication network of mobile nodes connected through radio links. The network is formed without any permanent or pre-deployed infrastructure.

MST Minimum Spanning Tree, a subset of arcs in a graph which connects the nodes in an acyclic manner, and which minimizes the total weight/cost of all the arcs in the tree.

OCSP Online Certificate Status Protocol, an application protocol used for clients to inquire about the revocation status of a public key certificate.

OLSR Optimized Link State Routing, a popular routing protocol for use in MANETs. Popular for its design of an efficient broadcast service, which otherwise tends to be an expensive operation.

P2P Peer-to-peer, a distribution principle where applications are built not based on client and server roles, but on symmetric cooperation between *peers*.

PKI Public Key Infrastructure, a set of services needed to create key pairs and certificates, distribute certificates and revocation lists and respond to online status queries.

POV Proof of Validation, a signed and timestamped statement about the validity of a certificate. Issued by the Validation Authority (VA).

SPST Shortest Path Spanning Tree. Same as Minimum Spanning Tree, but the tree is optimized for path length from any node to the root of the tree.

TCP Transport Control Protocol, a popular transport protocol protocol for connection oriented, reliable communication.

URL Uniform Resource Locator, a standard syntax for expressing an object retrievable from the internet. Often seen as the “web address” which can be used by a Web browser to retrieve information.

VA Validation Authority, an entity trusted by all nodes to judge the validity of certificates

XKMS XML Key Management Specification, an interface standard for a VA service

XML eXtensible Markup Language

References

[1] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume Books, April 2003.

[2] Tracy Camp, Jeff Boleng, and Vanessa Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, April 2002.

[3] Anders Fongen. XLMS Based Certificate Management, 2008/00278. Technical Report 2008/00278, Norwegian Defence Research Establishment, 2008.

[4] Anders Fongen, Morten Gjellerud, and Eli Winjum. A Military Mobility Model for MANET Research. In *IASTED PDCN'09*, Innbruck, Austria, February 2009. IASTED/ACTA Press.

[5] Anders Fongen, Frank T. Johnsen, and Eli Winjum. Certificate Validation in Military MANET based on Overlay Network of XKMS Proxies. In *MILCOM 2008*, San Diego, CA, USA, November 2008. IEEE.

[6] Anders Fongen and Terje Mikal Mjelde. A Simple MANET Emulation Testbed. Technical Report 2008/02290, Norwegian Defence Research Establishment, 2008.

[7] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk. RFC 5055 - Server-Based Certificate Validation Protocol (SCVP), 2007.

[8] Phillip Hallam-Baker and Shivaram H. Mysore. *XML Key Management Specification (XKMS 2.0)*. W3C, 2005. http://www.w3.org/TR/xkms2/.