FFI RAPPORT

EVALUATING THE PERFORMANCE OF JXTA OVER SATCOM

HAAKSETH, Raymond

FFI/RAPPORT-2004/02843

EVALUATING THE PERFORMANCE OF JXTA OVER SATCOM

HAAKSETH, Raymond

FFI/RAPPORT-2004/02843

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
P O Box 25, NO-2027 Kjeller, Norway

FORSVARETS FORSKNINGSINSTITUTT(FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2004/02843	UNCLASSIFIED	36
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFI-II/869/110	-	

4) TITLE

EVALUATING THE PERFORMANCE OF JXTA OVER SATCOM

5) NAMES OF AUTHOR(S) IN FULL (surname first)

HAAKSETH, Raymond

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH	IN NORWEGIAN
a) Middleware	a) Mellomvare
b) JXTA	b) JXTA
c) Performance	c) Ytelse
d) Disadvantaged grids	d) Radionett
e) Network Based Defence (NBD)	e) Nettverksbasert forsvar (NBF)

THESAURUS REFERENCE:

8) ABSTRACT

The transition towards a network-based defence brings along requirements to both network and distributed applications. Distributed applications are faced with a variety of networks ranging from high bandwidth links to disadvantaged grids. At the same time middleware has become one of the preferred programming models for such applications. Middware ease the development of distributed applications by hiding the complexity of distribution, and thus providing a new level of abstraction to the developer. Current middleware do however in most cases lack the ability to perform when running on low bandwidth and high latency links. In this report we evaluate the performance of the JXTA middleware when run over a satcom system with these two properties. We also make some suggestions on how to improve the performance.

9) DATE	AUTHORIZED BY	POSITION
5th November 2004	This page only Vidar S Andersen	Director

ISBN 82-464-0883-6

FFI-B-22-1982

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

CONTENTS

page

EVALUATING THE PERFORMANCE OF JXTA OVER SATCOM

1 INTRODUCTION

Access to information has become more and more valuable in todays and the future battle space. Traditionally information sharing has been executed in a centralised and planned manner. In the future Network Based Defence (NBD)[1], information sharing may be done in a more ad-hoc organisation [5]. To achieve this, a distributed information system that is capable of a dynamic configuration and reconfiguration is needed. This in turn yields the need for an underlying network that is capable of handling these types of systems.

At the same time middleware has become the preferred programming model for distributed systems, its goal in general is to support and ease the development of such systems. This is achieved by providing some level of transparency for the system developer. In addition some middleware provide platform, programming language and network layer independence. This is a crucial property when integrating different systems. Middleware has for some time now been in the focus of research at FFI. Several different middleware technologies have been investigated and the JXTA™ Peer-to-Peer (P2P) middleware has been chosen as the most promising at the moment. The choice of using JXTA is documented in [6].

To achieve the goal of dynamic configuration and reconfiguration of an information sharing system, the FFI project **NBD Decision Support** has developed a demonstrator of common operational picture compilation using Peer-to-Peer technology. This demonstrator is using the JXTA middleware as the platform of information sharing. This demonstrator is focusing on how peer-to-peer technology can be used to gain operational effect.

In this report we take a closer look at how JXTA perform when used in a disadvantaged grid[2]. During the NATO exercise Blue Game 2004 the above-mentioned demonstrator was deployed, as an experiment, in a real scenario. The experiment involved the FFI projects **NBD Decision Support**, **SIMUTREX** and **NBD Grid**. This report is focusing on the involvement of the **NBD Grid** project. To enable data exchange between sites, Iridium satellite communication was preferred for simplicity reasons. By deploying the demonstrator in a real life setting we have been able to evaluate how JXTA will perform in one type of disadvantaged grid. It is our hypothesis that the TCP protocol used to transfer JXTA messages may not be suitable for disadvantaged grids. Other more effective protocols should be deployed. Furthermore the JXTA protocols itself are to be investigated. It is believed that the amount of messages exchanged in a JXTA protocol session is too high and thus our second hypothesis is that the JXTA protocols are not suitable/optimised for disadvantaged grids.

The rest of the report is organised as follows. In chapter 2 the JXTA middleware is presented. Chapter 3 describes the experiment executed during Blue Game 04 in more detail. The emphasis

[1] NBD is the Norwegian equivalent for Network Centric Warfare (NCW)

[2] Disadvantaged grid is used to denote computer networks with properties like low bandwidth, high latency and high error rate. HF radio systems are one example of disadvantaged grids

is on the technical issues concerning the involvement of the **NBD Grid** project. In chapter 4 the topics of this evaluation of the JXTA performance is presented. This includes our focus and limitations of this evaluation. The results of this evaluation are presented in chapter 5. Finally, in chapter 6 a summary and conclusion on the evaluation are presented.

Conventions used in this document are twofold. First, some abbreviations and definitions are used. These are noted with the full name in the text with the abbreviation enclosed with parenthesis when first encountered. For the remainder of the document the abbreviation is used. Abbreviations that are assumed well known to the general public are not noted with full name. These and all other abbreviations and definitions can be found in Appendix A. Second, citings are listed in the References section and are enumerated. In the text, references are marked with numbers inside square brackets.

2 JXTA

JXTA is an open source middleware initiative originally provided by Sun Microsystems that is now being managed by the Project JXTA group [18]. JXTA is short for juxtapose which can be interpreted as equal or collocated. As the name suggest JXTA is a middleware platform for P2P programming. In opposite to the traditional client-server model, a peer in JXTA is likely to participate as both client and server. P2P systems are characterised by a high level of autonomy and heterogeneity, and no centralised control. JXTA defines a set of protocols for such applications to use for communication, these include protocols for discovery, group organisation and messaging. These protocols enable any networked device to communicate and collaborate. A networked device in this context is defined as any device connected to a computer network, this might range from small cell phones and PDAs to high performance PCs and servers.

P2P programming is not a new idea, in fact several applications have been built using the P2P paradigm. However, these applications have almost without exception used proprietary protocol solutions. As a result of this approach P2P applications from one vendor could not interact with applications developed by other vendors. Thus the benefits of P2P networks are limited to one, or at least a small set of, applications. The JXTA approach is different. By designing open and general purpose protocols any vendor can, in theory, create applications that can cooperate with applications created by other vendors. The specification enables JXTA to be independent from any platform, i.e. JXTA is not dependent on a particular operating system or programming language. The only requirement posed by JXTA is that it conforms to the JXTA protocols.

A reference implementation of JXTA has been implemented in the Java™ programming language using the Java 2 Platform, Standard Edition (J2SE). This implementation is available for download at [20]. Bindings to other programming languages are also available. Especially the C/C++ language binding is at the time of writing fully protocol compatible with the latest version of the Java reference implementation. Other language bindings are under development, this includes bindings for the programming languages Perl, Python, Ruby and Smalltalk. In addition a binding to the Java 2 Platform, Micro Edition (J2ME) are under development within the JXTA community group JXME [21]. J2ME is a distribution of the Java Platform targeted towards small devices like cell phones and PDAs. Other efforts to support mobile devices include a project to port the C/C++ language binding to Arm-Linux and WinCE. JXME should now be protocol compatible with version 2.0 of the J2SE implementation.

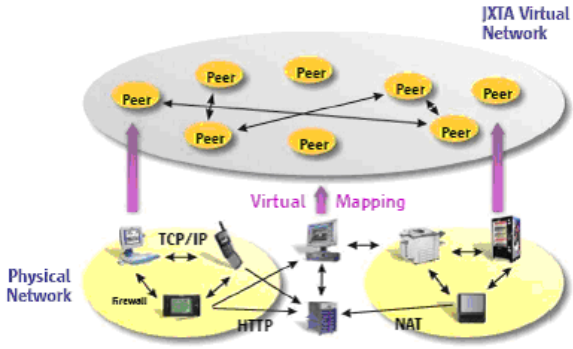

Figure 2.1: JXTA virtual network (source [18])

The basic idea of JXTA is presented in Figure 2.1. The participants in a JXTA session create a virtual network on top of the existing network by using the JXTA protocols. As mentioned above any networked and JXTA enabled device can be part of a JXTA session. A participant communicates with other participants with JXTA messages through JXTA pipes. Peers and peer services are discovered by the use of advertisements. In addition the JXTA protocols provide built in features for handling obstacles like Network Address Translation (NAT) and firewalls.

There is an abundance of information available on the subject of JXTA. Several books are available including [1, 8, 16, 24]. In addition the Project JXTA homepage [18] provides information on JXTA in general, including the JXTA protocol specification. More specific information on different projects within the Project JXTA community can also be found here.

2.1 P2P Concepts and JXTA Basics

P2P application and programming have its own set of terminology. The JXTA protocol suite follows these definitions. A peer as defined in [24] is *"Any entity capable of performing some useful work and communicating results of that work to another entity over a network, either directly or indirectly"*. Three different types of peers can be defined based on the properties they implement. These are simple peers, rendezvous peers and router/relay peers. Simple peers are what is traditionally known as a peer. This type of peer provides the user with the possibility to share or use resources within the rest of the P2P network. Rendezvous peers provide the capability of service and peer discovery by providing look up services. A simple peer can publish its services on the rendezvous peer or it can search for services of interest. A rendezvous peer may thus play the role of seeding in the creation of a P2P virtual network. In JXTA a peer publish or searches for a service by means of advertisements. Router/relay peers do also play a crucial part. They act as temporary storage of messages between peers that cannot communicate directly, e.g. peers behind firewalls and/or NATs. One peer might incorporate all of these types of peers. It is however most usual that simple peers are located at the edges of the P2P network and rendezvous and router/relay peers are placed more centrally and at known places, i.e. have a static IP address.

In JXTA peers can organise into groups called peer groups. In fact, every peer in JXTA is member of a group. If no group is specified the peer is a member of the default *World Peer Group*. Most

existing P2P applications are implemented with one problem in mind, the concept of groups have therefore not been developed. In JXTA on the other hand, groups can be created based on application domain, security requirements or the need for status information from the participating peers.

Services in a P2P system are defined as functionality offered by one or more peers to other peers in the network. Services can be partioned into peer and peer group services. One and only one peer within the network offers a peer service. This unique service will thus not be available when the peer in question is disconnected from the network. In opposite, a peer group service is offered by a peer group to other members of the group. That is, at least two or more peers within the group implement and offers this service to the other members. In this way some level of redundancy may be achieved in service access, i.e. if one peer offering the service disconnects other peers will take over.

2.2 JXTA Protocols and Specification

JXTA is essentially a set of protocols that is enabling peer-to-peer processing in a unified manner. The JXTA protocol specification [22] consists of six different XML based protocols. Of these, two are considered to form the core protocols. These are the Peer Resolver Protocol (PRP) and the Endpoint Routing Protocol (ERP). A peer must implement these two core protocols to be defined as a JXTA peer. In addition to be compliant with the JXTA specification a peer must implement JXTA IDs, JXTA advertisements, JXTA transport bindings and JXTA messages. The latter two are presented in separate sections, see sections 2.3 and 2.4 respectively.

JXTA IDs are used to uniquely reference entities situated in the P2P network. At the time of writing there are six different JXTA entities defined to have IDs. These are peers, peer groups, pipes, content, module classes and module specification. JXTA IDs are used to canonically, uniquely and unambiguously identify the resources within the scope of the P2P network. Other entities that must have IDs may be identified in the future.

In JXTA, advertisements are used to describe resources. Resources described by advertisements include, among others, peers, peer groups and services. Advertisements are language neutral meta data constructs that describe the service or resource. Programming language neutrality is achieved by the use of XML.

The Peer Resolver Protocol (PRP) and the Endpoint Routing Protocol (ERP) are, as mentioned above, part of the core specification. PRP is used to locate resources in the P2P network. PRP is designed for spreading generic queries within a peer group and for listening for replies. Queries and replies are addressed to a particular handler. Reply may come from one or several peers in the group, which implement the handler, in question. Resolver query and response messages and resolver SRDI query and response messages are part of PRP. The Endpoint Routing (ERP) defines a set of messages to help a peer route messages. This includes route query and response messages.

In addition to the core protocols described above JXTA defines a set of standard services and standard protocols which is optional [3]. The standard protocols include the Peer Discovery Protocol (PDP), Rendezvous Protocol (RVP), Peer Information Protocol (PIP) and the Pipe

[3]These services and protocols are optional but it is recommended that they are implemented for full conformance to the JXTA specification.

Binding Protocol (PBP). These protocols utilise the services and protocols from the JXTA core specification. PDP is used to discover resources in the P2P network that are represented by an advertisement. This protocol includes discovery query and response messages. RVP is used to propagate messages within a peer group. By using RVP a peer can propagate message to other peers and receive propagated messages. A peer running RVP identifies itself by issuing a Rendezvous advertisement. PIP, and messages defined by PIP, is used to query an identified peer for capabilities and status. Since PIP is optional a query might not be answered with a respond message. The last protocol PBP is used by peers to communicate with other peers and services. JXTA pipes and communication are explained below in section 2.3.

2.3 JXTA Pipes and Transport

Pipes are the main abstraction used for communication by JXTA. A pipe can be defined as a virtual communication channel used to transport messages from one peer to other peers. The JXTA specification [22] defines different types of pipes based on the Quality of Service (QoS) they may deliver. However, only uni-directional asynchronous pipes are required for a JXTA implementation. This type of pipe essentially provides the same functionality as UDP, i.e. messages are sent with no guarantee of delivery. Since this type of pipe is uni-directional, messages can only be sent one way, i.e. from one peer to another. Two-way communication between peers can be achieved by initiating two pipes, i.e. one in each direction. The argument for only requiring the uni-directional asynchronous pipe is that no possible network protocols are excluded, e.g. simplex communication. Other optional types of pipes defined in the JXTA specification include synchronous request-response pipe, bulk transfer pipe, streaming pipe and secure pipe.

The JXTA specification defines two different modes of communication. These are:

- Point-to-Point
- Propagate Pipe

The point-to-point mode provides communication between two different peers. In the J2SE reference JXTA implementation this is named JXTAUnicastPipe. The Propagate Pipe provides one-to-many communication, i.e. a multicast communication. In the J2SE reference implementation this is named JXTAPropagatePipe. In addition to these types of pipes, the reference implementation provides a secure unicast pipe named JXTASecurePipe. This type of pipe is essentially the same pipe as the JXTAUnicastPipe, but the connection is secured by using Transport Layer Security (TLS).

Since pipes are only an abstraction, some mechanism is needed to map this to lower level transport mechanisms. In JXTA this is done by the use of endpoints. Endpoints are the basic addressing functionality used by JXTA. An endpoint, as defined in [1], is *"an address of a peer that implements a specific protocol of communication"*. Endpoints are connected to pipes and are providing access to the underlying network interfaces used during communication.

As mentioned above JXTA endpoints provide access to the underlying transport mechanisms and underlying network interface. The JXTA specification defines three such message transfer

bindings. These are bindings for TCP/IP, HTTP and TLS. The reference implementation in addition provides message transfer for UDP and IP multicast when using propagate pipes.

2.4 JXTA Messages

Messages are the basic unit of data exchange between peers and services in JXTA. Applications and protocol entities exchange data by messages that are transported between peers using pipes and endpoints. A message is essentially a set of ordered message elements, where an element is a name/value pair. The content of a message element is arbitrary. The details of the message format are out of the scope of this report and are thus not presented here. Detailed descriptions can be found in [22, 16]. Messages can be represented in two different ways; XML and binary.

XML forms the basis of most protocols in JXTA and messages are also represented by XML. Using XML for defining protocols has both advantages and disadvantages. The main advantage of using XML is its flexibility and its ability to be validated. In addition, XML has become one of the most popular data definition languages, which has resulted in an abundance of both open source and commercial parsers. The popularity of XML also means that parsers are wide spread among users of computers. The main disadvantage of using XML is size. XML is not compact and thus creates large overhead when transported across a network connection.

To ease the problems of large overhead, JXTA defines a binary representation for messages. Binary message are more compact and more efficient for transport over a network connection. In addition since most messages are sent from application to application it is a trivial exercise to standardise the binary representation. Even though binary message representation is an alternative, the protocols are XML based.

3 EXPERIMENT

This section present the experiment performed during the exercise Blue Game 04 by the FFI project **NBD Decision support**, in cooperation with the FFI projects **SIMUTREX** and **NBD Grid**. The main objectives of the experiment were twofold. First, the experiment where aimed at exploring the potential operational value of ad hoc organisation of picture compilation nodes. Second, the demonstrator (see section 3.2 at page 14) was placed in an operational setting. The experiment is documented in several reports. In [23], the experiment and results in total are presented. The potential operational value is not investigated any further in this report, and the interested reader is referred to [10]. In addition, a synthetic environment provided by the FFI project **SIMUTREX** supported the demonstrator used in the experiment. This synthetic environment has been described in [9].

In this section we focus on the topics relevant to the involvement of the **NBD Grid** project in the experiment. This involvement included network support for the demonstrator and investigation of the performance of communication protocols. The network support is summarised in section 3.1. The technical setup of the demonstrator is presented in section 3.2. Analysis of the data communication in this demonstrator is used to evaluate the performance of JXTA. The sub experiment of investigating the communication protocols is described in section 3.3.

3.1 Communication

When making a choice of which type of communication to use during the experiment, the choice was limited to wireless carriers. This limitation is due to the placement of two nodes on-board vessels at sea. When considering different wireless technologies, four different criteria were considered; coverage, stability, data-rate/latency and ease of deployment. At the final stages three candidates were considered, these were:

- General Packet Radio System (GPRS)
- Globalstar Satellite Communication
- Iridium Satellite Communication

All three alternatives scored well on the deployment criteria. However, GPRS was ruled out since the coverage in the exercise area was considered insufficient. The two satellite communication systems Globalstar [7] and Iridium [15] were then the only alternatives left in contention. Both of these systems suffered from the same limitations. They both provided approximately the same data-rate and latency. Globalstar, did actually provide slightly better data-rate and lower latency. However, due to both better stability and coverage, Iridium was chosen as the better alternative. When testing these systems Globalstar proved to be unstable when exposed to heavy loads. Whether this was due to user errors on our behalf, or technical errors at Globalstar, is not known at the time of writing. Iridium did not experience such problems.

In addition to the wireless communication provided by Iridium, the NATO Unclassified LAN (NULAN) was used at the National Joint Headquarters (NJHQ). This was possible since the node at NJHQ was stationary. NULAN was used to connect the node to the public Internet.

3.1.1 Iridium

This section takes a closer look at the Iridium Satellite Communication system used in the experiment. Iridium deploys 66 satellites in low-earth orbit. Data communication can be achieved by either the Dial-Up Data service or the Direct Internet Data service. In the experiment the Direct Internet Data service was used, thus the rest of this section is concerning this service and for brevity it is just named Iridium. According to Iridium the Direct Internet Data service is capable of providing 10 Kbps data-rate.

During the selection process, the data-rate and latency of Iridium were measured using the Iperf 1.7.0 software [14] and ping respectively. Latency is defined as the round trip time (RTT) between two computers. RTT is defined as the time it takes to send a small data package from one computer to another and back again. These tests unveiled that the latency is considerable when using Iridium. The measured latency ranged from 1800 ms to 4000 ms between two computers connected to Iridium (two satellite hops). Between one computer and the Iridium gateway (one satellite hop) the latency was as expected approximately half of these values. It is important to note that when links are heavily loaded the delay will be higher. Data-rate measurement was also conducted to verify the figures promised by the Iridium specification. During our tests the average data-rate was 9,30 Kbps, with peak values of 0,80 Kbps and 21,40 Kbps. These test used the

build-in compression of the Apollo software distributed with Iridium. When testing without compression, the average data-rate was 1,2 Kbps. These results shows that the compression of data can improve data-rate considerably. The software compression was thus used during the experiment. The data-rate measured gives an indication of the data-rate encountered in some disadvantaged grids, or even a bit higher. It still provides a basis of our evaluation of the performance of JXTA. One issue that might effect the measurements is that Iridium use asynchronous links, i.e. the download data-rate is higher then the upload data-rate.

Iridium is using a NAT system for IP addressing. In essence this means that our computers connected to the Internet are using temporary and non-routable IP addresses. This posed a severe challenge and limitations for our experiment. First of all this means that no computers outside the Iridium network can communicate directly with computers inside. In addition, if a computer disconnects from the network it will receive a new IP address when reconnecting. This might provide routing problems within the applications. NAT is only one of the disadvantages of using Iridium. Another disadvantage is the fact that IP multicast is blocked at the Iridium gateway. IP multicast may provide more efficient method to disperse data to multiple receivers than point-to-point connections.

In addition to the technical aspects of bandwidth and latency, some minor issues should be mentioned concerning Iridium. Especially the accompanying Apollo Software used to connect to the Iridium network, i.e. the software used to setup a data connection. It has been known to be unstable when disconnecting and reconnecting without restarting the application. This is just a minor issue and Iridium as a whole has been working to our satisfaction.

3.2 The Demonstrator

The technology demonstrator used in the experiment was aimed to show ad hoc organisation on nodes participating in compiling a shared Common Operational Picture (COP). To achieve ad-hoc organisation, the J2SE reference implementation of the JXTA middleware was used (see section 2). JXTA is used to form groups that are cooperating in compiling a shared COP in an ad hoc manner. All communication between the peers [4] is performed by using the JXTA mechanisms. JXTA provides the ability for group members to join and leave without pre configuration. Internally each peer used several different middleware technologies like Jini and CORBA. Details concerning the demonstrator and its implementation is not within the scope of this document, and the interested reader is referred to [13].

The technical setup of the demonstrator used in the experiment is shown in Figure 3.1. Since Iridium use a NAT system, and the peer placed at NJHQ was situated behind a NAT system and a firewall, some measures had to be taken. This involved utilising the built in features of JXTA for handling such challenges. By deploying a rendezvous/ relay peer at FFI, at a known and stable IP address, the peers could communicate indirectly via this peer. During the experiment, the demonstrator used the HTTP Transport binding of JXTA. That is, all communication was using the HTTP protocol on top of TCP. TCP ensured reliable data transfer. The multicast option of JXTA was not a viable option since Iridium blocks such traffic.

In addition to the physical setup of the demonstrator, some measures were taken to adapt to the

[4] Since JXTA is used, the expression *peer* is used instead of *node* for the rest of this document.

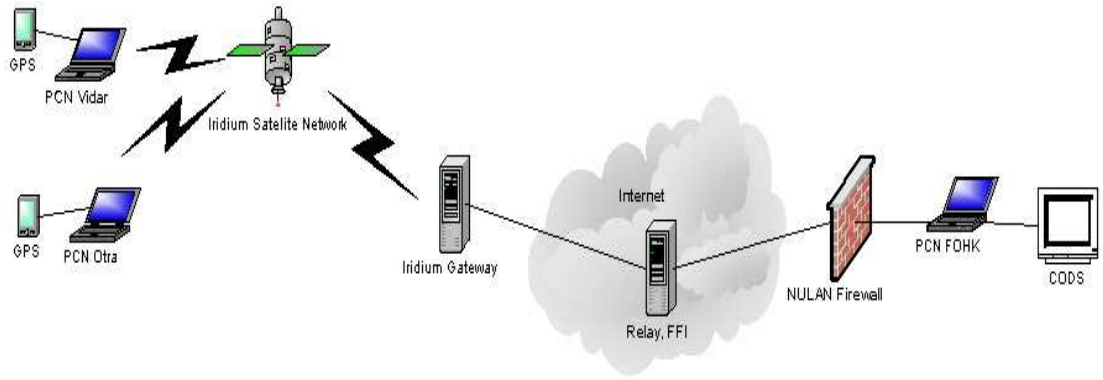

Figure 3.1: Technical setup of the Blue Game Demonstrator

low data-rate of Iridium. Until this experiment, the demonstrator had only been executed in a Local Area Network (LAN) environment. The most notable adaptation made was the use of larger messages. To avoid time consuming connections and extra overhead, messages were buffered and concatenated into one message which was sent at a given time interval.

3.3 NBD Grid Experiment

In order to evaluate the JXTA and transport protocols used by the demonstrator, we logged the network traffic generated by the demonstrator. The actual logging was performed by using Ethereal [4]. This is a network protocol analyser software capable of sniffing and storing packets from the network interface. Since all communication between peers in the demonstrator was performed via the rendezvous/relay peer, it was only necessary to log traffic arriving at this peer and one node using Iridium for communication. The Iridium enabled peer was chosen since we were interested in evaluating the performance of JXTA in low data-rate networks.

4 ANALYSIS

When evaluating the performance of JXTA over Iridium, there are two aspects to consider. The first is the transport protocols used, in this case TCP. The other aspect is the protocols defined and used by JXTA itself. Since JXTA is a relatively new technology, there has not been much work done in the area of defining the performance of JXTA. Nevertheless some efforts has been undertaken, this include work done at the University of Saskatchewan [11, 12]. In addition, project JXTA has realized the need for better understanding of its performance. A sub project has been initiated that is to provide a repository of benchmarks for performance testing [19].

4.1 Proposed Performance Model

In [11] a performance model for evaluating JXTA implementation is proposed. This model consists of the five elements:

1. Typical Peer Operations
2. Pipe Message RTT
3. Message and Data Throughput
4. Rendezvous Query Throughput
5. Relay Message Throughput

The first element corresponds to the cost of starting a peer. This involves operations ranging from loading the JXTA libraries and checking the advertisement cache to joining peer groups, publish advertisements, open pipes and to discover other peers and services. In addition, operation involving initial rendezvous and relay peers connections are included. The second element, pipe message RTT, corresponds to the time it takes to send a message and receive an ACK for that message. Since messages are the basic communication entity in JXTA it is used instead of e.g. transport packages. Message and data throughput does, as the name suggest, deal with the maximum number of messages that can be sent in a pipe without loss. The last elements are concerned with the implementation of the rendezvous and relay services respectively. This involves measuring the query response time and throughput of messages. These last elements are important since both rendezvous and relay peers are believed to be exposed to large amount of traffic.

This performance model has been used to evaluate the performance of JXTA and the results are presented in [11, 12]. At the time of writing results from the three first elements in the performance model have been published. In addition there is work in progress on evaluating the last two elements. It should be noted that these results are somewhat deprecated since they are based on an older version of the JXTA J2SE reference implementation than the one used in the demonstrator. Nevertheless the results presented are of interest.

The results are not presented in this report, but they serve as an example of work done to investigate the performance of JXTA.

4.2 Objectives of Evaluation

All the efforts described in section 4.1 have in common that the tests are all performed in identical environments, i.e. running on a high data-rate LAN. Although interesting, these are not representative for our evaluation. When evaluating the performance of JXTA over satellite communication we consider different topics. While the other efforts have concentrated almost solely on higher-level analysis, we focus on both the JXTA protocols and the underlying transport protocol, which in this case is TCP. In our analysis we have considered these topics:

- Number of Connections
- Duration of Connections
- Connection setup time
- Retransmissions

- TCP payload
- Peer Discovery
- Pipe setup
- JXTA payload

The first five items concerns the underlying TCP transport protocol. The number and duration of connections is interesting since, in terms of TCP, many short-lived connections are more expensive then few long-lived connections. This is due to the fact that TCP requires an initial setup, and termination using a three-way handshake. This is expensive and that will become even more so if many short lived connections are used. In addition the TCP congestion and flow control mechanisms should in principle make the sending of data come to a pivot when the amount of errors are minimal. Thus, in theory, long-lived connections should make for better utilisation of the available data-rate. The third subject of our investigation, connection setup time, is closely related to the above-mentioned number and duration of connections. In high data-rate and low delay networks this value is almost negliable. However, in disadvantaged grids with low data-rate and high delay the time used to setup a connection might be considerable. Other interesting topics of our investigation include TCP payload and the number of retransmissions. The number of retransmissions is as the name suggests the number of times a TCP packet must be retransmitted due to a timeout when waiting for the ACK. The TCP payload metric is used to evaluate how much application data versus how much protocol data that are transported, i.e. finding protocol overhead in terms of packets and number of bytes.

The last three items considered in this evaluation concerns the JXTA protocols. In order to evaluate the performance of JXTA, we need to say something about the amount of payload in each message. This corresponds to the JXTA protocols and how much of the data transferred is actual application data. In addition the JXTA mechanisms for service discovery and pipe setup are examined. These two metrics are important for establishing a picture of how the internals of JXTA perform at the middleware layer.

It is claimed that the JXTA protocols are designed to have very low overhead, make few assumptions about the underlying network transport and impose few assumptions on the peer environment (see [22]). It is the intention of this evaluation to use the listed topics to verify this assertion.

4.3 Not Considered

In addition to the metrics considered in this evaluation, others may also have been evaluated. The reason for omiting these is first of all that they are not believed to have profound effect on the total performance of JXTA. Secondly, many of the JXTA specific performance metrics are connected to abstract functionality only visible at the middleware or application level. The network logs used in this evaluation is thus not suited for evaluation of such abstract functionality. Use of the proposed performance model (see section 4.1) and logging at this particular level is the best alternative for measuring these.

Issues not included in this evaluation come from both the transport and JXTA protocol layer. At the transport level the list could have been expanded to include the evaluation of time used to

terminate a TCP connection. Like the connection establishment phase, the termination phase also uses a three-way handshake. This metric is interesting since this process may occupy the communication channel and hence delay other, perhaps more useful, data. It is believed that this is as influential as the connection establishment, and the overhead can thus be deducted from the results of this metric and is hence dropped in this evaluation. Issues like round-trip time (RTT) and throughput of both TCP and JXTA pipes are not considered, these metrics, although important, are not considered in this evaluation. It is believed that to measure these, a benchmark that can stress the protocols to a maximum is needed. The demonstrator is not such a benchmark (see section 4.4). The payload ratio of both TCP and JXTA protocols are instead used to measure the real throughput of application data.

4.4 Unknown Factors

This evaluation is based on logs of the network traffic generated by the JXTA enabled demonstrator using satellite communication from Iridium. Neither of these can guarantee the same level of service at different times. The demonstrator is fairly stable and will generate approximately the same amount of data and messages to be sent between peers in each run. The results should thus be comparable. This can however not be guaranteed since the demonstrator used is not a controlled benchmark, e.g. messages in the demonstrator can be of different sizes from one run to another. The demonstrator used in the experiment was implemented to send messages at a given time interval. At one interval all available data, i.e. track data used to establish the Common Operational Picture (COP), was sent to the other peers connected. And at a shorter interval only data that was updated since the time of the last sending was sent. Due to variations in the total amount of data and updated data, the output generated by the demonstrator may vary over time; hence results may be affected by this factor.

Finally, the fact that the number of the logs is fairly small, results in the fact that the variance of the results become larger, and thus more influential on the overall results. The foundation for making an absolute conclusion is to weak and hence the results presented in section 5 must be considered with this in mind. To attain more certainty in the results, a benchmark software that has the property of reproducing the same amount of data at the same time for each execution must be used. In addition, this benchmark must be used in a controlled communication environment, e.g. in a lab. Even though the results presented here incorporate a certain degree of uncertainty, they serve as a good example of the performance of JXTA when placed in a disadvantaged grid like the Iridium network.

5 RESULTS

In this section the results from the evaluation of the network traffic logs from the Blue Game 04 experiment are presented. This section will look at the JXTA performance over the Iridium link. In addition, this section will try to identify problem areas. This section is divided in three. In section 5.1 the empiric results are presented. These results present some of the experience we have gathered during the experiment that cannot be measured in a formal matter. In section 5.2 and 5.3 the results of the evaluation of the transport binding and JXTA protocols are presented.

To generate JXTA traffic for the analysis two peers were used, one Iridium enabled peer and one peer connected to the NULAN at NJHQ (see figure 3.1). These two peers communicated via the rendezvous/ relay peer. In total, ten logs were captured during the experiment. These logs are distributed over five runs of the demonstrator. For each run the network traffic was captured at both the rendezvous peer and at one of the Iridium enabled peers. Results may vary between runs due to the fact that these logs are captured using a software, i.e. the demonstrator, that cannot be guaranteed to provide the same output on different executions. In addition, the communication environment, i.e. Iridium, can not be controlled. The fact that the length in time of logging data is different is also contributing to the difference in the results achieved. Synchronising the logs at two different locations also posted a problem, this may also have resulted in differences between the two logs of one capture.

5.1 Empiric results

Much information and knowledge can be gathered by just using the demonstrator. Even though this knowledge cannot be measured in a scientific way it is important to note the perceived performance of JXTA. This section takes a closer look at the hands-on experience of using the demonstrator during the Blue Game 04 experiment. Among the lessons learned by the experiment is the importance of configuration. During the preparation of the experiment, several different configuration alternatives were tested. The results of testing these alternatives varied from not working at all to the setup presented in section 3.2, which worked to our satisfaction. In addition, the buffering and sending of messages in larger packets proved to be a valuable asset.

Despite the fact that the demonstrator and JXTA, all things considered, worked as expected, some exceptions should be noted. First, the exchange of advertisements was very slow, even slower than expected. In average the exchange of advertisements between two peers, when both were connected via Iridium, was in the order of minutes. This is clearly not acceptable. In addition, during runtime the demonstrator often seemed to slow down, i.e. it did not produce or send messages for some time. In these cases the demonstrator had to be restarted. This behaviour was not in accordance to the implementation of the demonstrator. As of the time of writing the reason for this error is not found. It is however suspected that a timeout value in the J2SE reference implementation of the HTTP/TCP transport binding caused this problem. During testing over GPRS, which has considerable better data-rate, and most important lower latency, we did not experience the same problems. Iridium itself is not a suspect of these errors since the communication did not fail.

5.2 Transport protocols

The chosen transport protocol for the demonstrator is HTTP version 1.1 over TCP. Actually HTTP is classified as an application level protocol, however in JXTA the use is more similar to a transport protocol. HTTP is a stateless protocol that provides a convenient way of transporting application data. Nevertheless, when analysing the transport protocol of the demonstrator the point of interest is TCP. It is believed that this protocol will provide the most extra overhead. This is due to the fact that TCP is a connection oriented protocol that provides reliable data transfer. The fact that TCP is connection-oriented leads to the need for a connection establishment and

Capture	Connections	Total time	
1 a	39	22 min	7 sec
1 b	38	21 min	36 sec
2 a	30	25 min	42 sec
2 b	30	23 min	12 sec
3 a	29	20 min	37 sec
3 b	31	20 min	6 sec
4 a	32	26 min	33 sec
4 b	32	26 min	31 sec
5 a	32	30 min	28 sec
5 b	32	30 min	27 sec

Table 5.1: Number of connections used.

termination. In the case of TCP this is achieved using a three-way handshake. Reliable data transfer, although useful in many respects, may introduce overhead not suited over a disadvantaged grid. This overhead is introduced by the need for keeping track of connection management information and unnecessary retransmission of packets. In the following sections the results of our evaluation of the issues identified in section 4.2 are presented. The results are presented in tables where each row represents the result from evaluating the log with number indicated by the *Captures* column. Captures marked with an *a* correspond to captures done at the rendezvous peer and captures marked with a *b* represents the Iridium enabled peer.

It is important to note that the results presented in this chapter are based on what was observed during the experiment. They are based on the use of the default TCP implementation available in the Windows XP Professional operating system. This implements the “Reno” version of TCP. No efforts were taken during the experiment to tune the parameters of this implementation to gain performance advantages. Although tuning may result in performance gains, this has not been within the focus of this evaluation.

5.2.1 Number of Connections

The number of connections used is an important metric on how the protocol performs in a disadvantaged grid. This metric is actually not concerned with the TCP protocol as such, but rather how the higher level protocols, i.e. JXTA in this case, utilise TCP. This metric must be seen in relation to the next metric, duration of connections. Often fewer long-lived connections are more economic in terms of both throughput and startup latency.

Table 5.1 summarise the results from analysing the logs. In average each execution of the demonstrator use about 30 connections, with the exception of the first run represented by capture *1 a* and *1 b*. In addition the table contains the total time of logged data traffic in each capture. In average, the demonstrator/ JXTA created one new connection every 45 seconds. In order to decide how effective this is the result must be seen in comparison with the other metrics mentioned above.

5.2.2 Duration of Connections

The duration of a connection can be measured in two ways. First the actual time a connection is active can be measured. Alone this metric does not provide much useful information. One may use this to conclude that a connection is either long or short lived, i.e. if the connection may be used to send large or small amounts of data. The amount of data sent is the second metric that can be used to measure the duration of a connection. More specific the number of packages sent during the lifetime of a connection. In this evaluation a combination of these is used to denote the duration of a connection. By doing this both the actual time and the degree of utilisation can be incorporated into the duration of a connection.

Table 5.2 summarises the duration of connections in seconds. This table do only show the minimum, maximum and the average duration in time and number of packages. Similar Table 5.3 on summarises the utilisation of the connections in terms of packages with minimum, maximum and average values.

In terms of time, the connections ranged in duration from as short as eight seconds to almost 1800 seconds, or almost 30 minutes. In general what can be seen from the logs captured is that a JXTA session uses a small number of long lasting connections. These connections are often open and active through the lifetime of the execution. In addition, the logs show that a set of medium and short-lived connections are used. Actually the main portion of all connections falls into this category. The average connection time for each log varies from about 142 seconds to about 246 seconds, which is approximately from 2 to 4 minutes.

When measuring the utilisation of the connections in packets instead of time we found, much as expected, the same pattern. The utilisation of the connections vary from as low as 10 packages as a minimum till about 650 packages in one connection as the maximum. The average amount of packages transported in a connection varies from about 50 to 74 between the different captures. Naturally, there is a correlation between the connections that are long lived in time and number of packages sent and received.

Combining the duration and utilisation metric provides us with the knowledge of how the connections are utilised. It does make not much sense of keeping a connection live if no data are sent. In the case of the demonstrator and the JXTA middleware, only a few of the long lasting connections were actually used for sending large amount of data. The rest is used for what seems like keep-alive traffic. This use of long lasting connections is not effective. The short connections are used for transport of packets during the whole lifetime, and hence the connections are well utilised. However, this data could have been transported in the connections that are established and not used.

5.2.3 Connection Setup

This is the first metric concerned directly with TCP and how it behaves over Iridium. TCP uses, as mentioned above, a three-way handshake at the establishment and termination phases of a connection. We will only focus on the connection establishment phase and anticipate that the connection termination takes the same amount of time. As the name suggest, this involves three messages. The first message is a synchronisation message sent from the initiator, denoted client,

Capture	Min	Max	Average
1 a	16.1254	1307.915	142.5863
1 b	16.22497	1294.069	414.4338
2 a	15.771	1542.339	245.6274
2 b	8.288434	1392.069	221.1563
3 a	16.07701	1225.596	224.4842
3 b	7.949124	1186.201	220.7359
4 a	15.70479	1321.687	228.0356
4 b	15.57118	1356.239	230.8046
5 a	16.26939	1797.353	226.4415
5 b	16.31448	1799.284	227.0264

Table 5.2: Duration of Connection in seconds

Capture	Min	Max	Total	Average
1 a	14	499	1852	50.611
1 b	14	481	1818	49.667
2 a	14	557	1942	70.926
2 b	10	543	1917	68.889
3 a	15	441	1768	67.875
3 b	10	437	1790	66.833
4 a	16	518	2043	74.000
4 b	16	527	2109	74.592
5 a	12	649	2115	70.000
5 b	12	618	2132	69.833

Table 5.3: Utilisation of the Connections in number of packets

to the receiver, denoted server. The server acknowledges this message by sending an ACK message. In the same package the server sends its own synchronisation information. The client acknowledges the reception of this package with the third and last message. The connection setup time is measured in two different ways in this analysis, dependent on whether the log is representing the client- or server side. For the server this time is calculated from the arrival of the first package to the arrival of the last acknowledgement message. The setup time at the client side is calculated from the time of sending the first synchronisation message to the time of sending the last acknowledgement message. In the demonstrator, and hence also the captures, the Iridium enabled peer will always be the client and the rendezvous peer will be the server. By measuring the connection setup time this way the results may vary. It was however, as mentioned above, difficult to synchronise the logs for a more precise measurement. In addition, Ethereal measures time for the capture of the first package, whether it is transmitted or received. It does not measure the time from the start of the sniffer, it is thus impossible to synchronise the logs due to the fact that we cannot control when the first package is sent or received.

In Table 5.4 the minimum, maximum and average time used for setting up a connection on each log is summarised. The time used to setup a connection is substantial. For the client and server combined, the time varies from 1.26 to as large as 23.99 seconds. On the server side the time varied between 1.28 and 12.27 seconds. And on the client side this varied from 1.26 to 23.99 seconds. In average the time used to establish a TCP connection is between 2 and 3.5 seconds. Based on the latency values (see section 3.1.1) these values are as expected. Establishment of a TCP connection requires one round trip and one additional message, i.e. one and a half round trip is used. The minimum and average results are in accordance with a theoretic time calculated from the minimum and maximum measured round-trip time. The maximum values are however not in accordance to this. It should be noted that only a small number of the connections use this much time connecting. Nevertheless, such high values must be taken into consideration when deploying a transport protocol in a disadvantaged grid. The cause of such high delays can be lost or delayed packages.

In addition it should be mentioned that a small number of the connections failed during each run of the demonstrator. The results of these connection attempts are not part of the results presented in Table 5.4. The cause of these failures is not known at the time of writing. It is however believed that it is a combination of JXTA, Iridium and a firewall in front of the rendezvous peer.

5.2.4 TCP Payload

The payload metric is concerned with how much of the total data transmitted is actual application data and how much is protocol information. In this section we concentrate on the transport protocol TCP. Thus, application data is defined as HTTP data, JXTA protocol data and application data from the demonstrator. The reason for investigating this metric is that we can gain more insight into how much protocol overhead is introduced by TCP itself. Similar to the duration of connections metric, the payload metric can be expressed in two different ways. In this evaluation we first use the number of packages containing data, and second how much application data is sent in bytes.

Table 5.5 presents the amount of packages containing application data in the logs captured. From this table we can see that less than half of the packages captured contain application data. The

Capture	Min	Max	Average
1 a	1.342882	7.006446	2.22017717
1 b	1.357029	23.990461	2.87650086
2 a	1.353075	9.917902	2.46921104
2 b	1.376974	15.771176	2.80779226
3 a	1.343053	7.882393	3.06026756
3 b	1.381193	14.939467	3.53609324
4 a	1.278901	12.269773	3.44818907
4 b	1.264307	14.887015	3.76880622
5 a	1.294982	7.927234	2.6615265
5 b	1.356076	5.613569	2.78123873

Table 5.4: Time (in sec) used to setup a connection

average amount of data packages vary from constituting 44,71 % of the total volume of packages to 48.82 %. Thus, in terms of the number of packages TCP generates an overhead of over 50 percent that are of no use for the application directly. This number is substantial when considering the low bandwidth and high latency experienced when using a disadvantaged grid like Iridium.

In Table 5.6 application data payload is shown in bytes. In this table we see that despite the fact that over half of the packages sent do not contain application data, the application data constitute a big part of the bytes sent. In fact, over 90 % of the packets is data. The lowest percent of application data was 92.82 % and the highest registered was 94.08 %. The values presented in Table 5.6 include whole packages with TCP, IP and Ethernet headers. Thus, this table does only say something about the difference in packet size for packets containing application data versus packets that do not. This does not enable us to directly say something about how much data is actually sent. A rough estimate can however be calculated by finding the average bytes used in a non-data packet, i.e. the TCP overhead per packet and reducing the bytes used in a data packet with this value. This is a rough estimate since it does not take into consideration what header fields of TCP that are used. Nevertheless, such an estimate shows that still about 85 to 90 percent of the bytes sent is actual application data.

It is also worth mentioning that during the evaluation we found, as expected, that connections lasting for longer periods are more effective. Shorter connections had a percentage of packages containing application data as low as 35 to 40. This number was as high as 53 percent for longer connections.

5.2.5 Retransmission

This metric says first of all something about how many packets are lost and secondly how many packets that could potentially be lost if TCP or another reliable transport protocol had not been used. It should be specified that a TCP packet is retransmitted if it is believed lost by the sender, that is when its retransmission timer expires. The packet in question may in fact not be lost, the packet or the corresponding ACK may potentially only be delayed in the network. The setting of the timeout value is thus of extreme importance. To reduce the amount of unnecessary

Capture	Packages	Data packages	Percent of data packages
1 a	1852	828	44.71 %
1 b	1820	819	45.00 %
2 a	1942	948	48.82 %
2 b	1917	934	48.72 %
3 a	1768	828	46.83 %
3 b	1790	848	47.37 %
4 a	2043	952	46.60 %
4 b	2109	993	47.08 %
5 a	2115	1003	47.42 %
5 b	2132	1016	47.65 %

Table 5.5: Payload of the connections in number of packages.

Capture	Data bytes	TCP bytes	Total bytes	Percent of data bytes
1 a	771332	59650	830982	92.82 %
1 b	756563	55786	812349	93.13 %
2 a	875804	57878	933682	93.80 %
2 b	871977	54846	926823	94.08 %
3 a	744841	55184	799998	93.10 %
3 b	764163	53016	817179	93.51 %
4 a	856136	63750	919886	93.07 %
4 b	891738	62416	954154	93.46 %
5 a	942538	64416	1006954	93.60 %
5 b	963385	61900	1025285	93.96 %

Table 5.6: Payload of the connections in bytes.

Capture	Packages	Retransmission	Percentage
1 a	1852	19	1.03%
1 b	1820	70	3.85%
2 a	1942	32	1.65%
2 b	1917	116	6.05%
3 a	1768	25	1.41%
3 b	1790	133	7.43%
4 a	2043	28	1.37%
4 b	2109	110	5.22%
5 a	2115	19	0.90%
5 b	2132	72	3.38%

Table 5.7: Retransmissions of packets

retransmissions, TCP sets the timer to an estimate of the RTT. This estimate is calculated using a exponential weighted moving average (EWMA) of the RTT. This ensures that the latest RTT measured from the time of sending a packet to receiving the ACK is weighted the most in the average. In addition fluctuation in the network is considered when estimating the RTT. Thus the current state of the network is reflected in the timeout.

In table 5.7 the results of measuring the amount of retransmission during the experiment are presented. The number of retransmission may vary from under one percent of the total number of packages sent to over seven percent. This last figure is high and it serves as a good example of how a reliable transport scheme like TCP, using positive ACKs, may end up with degrading the potential of a network link. Retransmission of packets in itself may cause congestion on the link, which may trigger even more timers expiring leading to even more retransmissions. This again results in an increase in latency of sending new data, due to the congestion and flow control mechanisms in TCP. This may seriously hamper the performance of applications with real-time requirements.

When evaluating the amount of retransmissions it is worth noticing that the Iridium enabled peer (captures marked with *b* in Table 5.7) retransmits considerable more packets than the rendezvous peer. While the number of retransmissions on the rendezvous peer constitute from less than one percent to less than two percent of the total number of packages, the number of retransmissions at the Iridium enabled peer range from over three to over seven percent. The reason for this noticeable difference might originate from the fact that Iridium uses an asynchronous link with better download than upload capabilities. The delay due to the asynchronous link is more noticeable when the Iridium enabled peer is sending data, thus the higher retransmission rate.

5.3 JXTA Protocols

According to the specification [22] the JXTA protocols are designed to have very low overhead and assume as less as possible on the underlying network transport and peer environment. According to this, JXTA should perform well in a disadvantaged grid environment like the one encountered when using Iridium. This section takes a closer look at the results from evaluating the performance of the JXTA protocols when placed in a disadvantaged grid.

One overall observation made during the evaluation, is that the JXTA protocols are very chatty. By chatty we mean that many messages are exchanged between the peers that are communicating. In terms of amount of packages sent, the JXTA communication corresponds to the number of packages with data presented in section 5.2.4 and in table 5.5. In the same section the amount of JXTA data is summarised in table 5.6 as data bytes.

This section is partitioned into three different sub sections, first the time used to discover other peers are presented in section 5.3.1. Second, the time used to connect to the other peer, i.e. time used to setup a pipe, is investigated in section 5.3.2. Last, but perhaps most important, section 5.3.3 presents the payload ratio when using JXTA. The results presented here are based on an evaluation of the logs captured at the Iridium enabled peer. In total this constitutes five different logs.

5.3.1 Peer Discovery

The peer discovery metric is used to measure the effectiveness of the discovery service. In this evaluation it is defined as the time it takes to discover that the other peer is active during the experiment. This includes finding and downloading the advertisement of the services from the other peer. Advertisements in this case, are to be interpreted as the Module Specification Advertisement (MSA), which describes the available service, including information on how to connect to it. Since both peers are started at approximately the same time there should be no difference in the timing as to when discovery is started. There are however some factors that may influence the measurement, these were discussed in section 4.4. The uncertainty listed here effects the results of service discovery more severely since for instance a failure of Iridium may delay the sending of advertisement considerably. In addition, since the number of samples is so low, the effects of these types of variations become more influential on the results. Alternatively, the number of packages, or the amount of data that is exchanged before a successful discovery has been made can measure peer discovery. This way of measuring peer discovery is not elaborated further in this evaluation.

Table 5.8 summarises the time used to discover the services available in the JXTA network created for the demonstrator. During the execution of the demonstrator for the NBD Grid sub-experiment only two peers were active. These were the Iridium enabled peer situated on board HNoMS Otra, simulating HNoMS Bergen, and the NULAN enabled peer situated at NJHQ, simulating a MPA (see figure 3.1). The columns of Table 5.8 show the time it takes for the Iridium enabled peer to discover the services and download the advertisements of these services for the first time. Since both peers are started at the same time and the advertisement cache is empty for all peers, including the rendezvous peer, the figures presented will also include the time used to upload the advertisements to the rendezvous peer.

As we can see from the table, service discovery takes a considerable amount of time. The first thing to notice, is that discovery of services located on the same Iridium enabled host takes longer time than discovering remote services. On average it took almost four minutes to discover the service located on the NULAN enabled peer (featured in the MPA column of the table), while it on average took seven minutes and 40 seconds to discover the services located on the peer itself. The suspected reason for this is the extreme difference in data-rate experienced by the respective peers. It hence takes longer time to upload an advertisement from the Iridium enabled peer. The

Capture	MPA	BERGEN
1	5m 23s	6m 19s
2	1m 35s	7m 12s
3	1m 48s	7m 45s
4	6m 42s	10m 44s
5	3m 46s	6m 21s
Average	3m 51s	7m 40s

Table 5.8: Time used to discover services from other peers.

asynchronous link of Iridium also plays an important role in slowing down the discovery of services on the Iridium enabled peer. The reason why own services are discovered is due to an ineffective implementation of the JXTA discovery service, which do not filter out services situated on the query origin peer. This combined with a perhaps less then optimal construction of queries.

The time used for discovery is, as mentioned, considerable, one reason for this is the size of discovery queries, discovery responses and advertisements to be exchanged between peers. With this setup of the demonstrator, a single discovery query took three TCP packages. In terms of bytes to be transported across the disadvantaged Iridium grid, this sums up to 4034 bytes, which include TCP, IP and Ethernet protocol data. An advertisement of type MSA is approximately 2000 bytes in size dependent on different factors such as length of service names. The MSA is incorporated in a discovery response message, which provides an additional amount of bytes, approximately 1600 bytes. A MSA occupies on average three TCP packages and the discovery response message covered one and a half TCP package. This figure do vary dependent on the number of MSA advertisement incorporated. IP package size was during the experiment set to default 1500 bytes. Having the relative low data-rate of Iridium and the results from evaluating the transport protocol (TCP) used in mind, the service discovery have a relative poor performance. It should also be mentioned that during this sub-experiment a relatively low number of services were available. Scaling the demonstrator to more services will effect the discovery functionality in a way correspondent to the number of advertisements that must be uploaded and downloaded.

5.3.2 Pipe Setup

The pipe setup metric is in this evaluation used for considering the effectiveness of the pipe abstraction. The demonstrator was configured to use the JXTAPropagatePipe, which provides one-to-many communication (see chapter 2.3). The chosen transport mechanism is however HTTP over TCP/IP, which provides connection-oriented unicast. How this difference is solved internally by the JXTA J2SE implementation is not known at the time of writing. In addition, since pipes are an abstraction used by the JXTA middleware itself, and the applications using the middleware, pipes are only visible at these levels. Due to this fact the effectiveness of pipes are best measured at these levels, and not by low level communications logs as those used in this evaluation. It is difficult to say something exactly about the time used to perform a setup of a pipe when investigating communication logs, due to the fact that pipes are only virtual communication channels.

Capture	Pipe Setup Time
1	2 min 07 sec
2	7 min 25 sec
3	9 min 27 sec
4	5 min 27 sec
5	3 min 59 sec
Average	5 min 41 sec

Table 5.9: Time used to setup a JXTA pipe

Pipe setup is defined as the time used from a service is discovered until the two peers can communicate, i.e. until a pipe is created and is ready to be used. Due to the limitation mentioned above, this evaluation must use a more relaxed definition of the pipe setup metric. In this evaluation the pipe setup is defined as the time from receiving the MSA advertisement, i.e. discovering the service, until the first application data is received from this service. This definition is somewhat inaccurate and provides results that are of lower precision than application level measurements. Nevertheless they serve as a good representation of what the actual pipe setup time is. It should however be noted that the results are affected by the fact that no data is produced by the demonstrator until it is stimulated by the synthetic environment, hence results may vary due to variations in start time of this.

Table 5.9 shows the results from evaluating the network logs, when using the definition mentioned above. The results presented in this table are based on the time difference from receiving the advertisement from the remote peer (i.e. in this case the MPA peer simulated from NJHQ) until receiving the first data from this peer. On an average this process took more than five minutes, which is a considerable amount of time. It should also be noted that there is a relation here between discovery and pipe setup time. When the total discovery time (i.e. receiving the MSA of both the remote and local peer) is large the pipe setup is also large. This may be due to the previously mentioned fact that the startup procedure involved not sending data before all pipes were established. Hence when the Iridium enabled peer had problems publishing its own advertisement it will slow down the whole process.

The results presented in this section are far from accurate and should not be used to make conclusion on the effectiveness of the JXTA pipe abstraction. From the results presented, it is very difficult to compare the performance of pipe setup and the Pipe Binding Protocol with other JXTA protocols. It is thus not possible to deduce how this JXTA protocol performs when compared with others. The results of this metric are only presented as is; they are not intended to be used further due to the uncertainty associated with them.

5.3.3 JXTA Payload

In section 5.2.4 the payload ratio of TCP during the experiment was presented. The results presented do however not consider the fact that the JXTA protocols provide additional overhead in themselves. This section presents the results of evaluating the perceived application data payload ratio of JXTA during the experiment. It is difficult to define exactly what is payload when

using JXTA. This evaluation does only consider data produced by the application part of the demonstrator as payload. In this case, data is defined as track data produced by the demonstrator. That is, all JXTA protocol data is considered as overhead. The JXTA protocols may themselves perform useful operations, however when considering the payload ratio all protocol information must be stripped. HTTP protocol information accumulated due to the use of this protocol is also included in the JXTA overhead. This overhead is extremely small in comparison with the JXTA protocol overhead and will thus not affect the final results.

Table 5.10 summarise the results from analysing the logs captured during the experiment. This table does only feature the results from the Iridium enabled peer. The argument for only showing these results is that the payload ratio will be the same at either end, since they are not dependent on time measurements. However, due to the fact that packets containing data may be retransmitted there may be some variations. As we saw in section 5.2.5 the Iridium enabled peer had a higher percentage of retransmissions than the Internet enabled rendezvous/ relay peer. Retransmission of both application and protocol data is included in these figures.

The table summarises both received (input) and sent data (output). Data is received via the relay peer. As described earlier this peer acts as a router and forwards data to peers behind firewalls or NAT systems. Data from the relay peer are transmitted in one connection that is kept alive throughout the execution. The *Input* column of Table 5.10 feature the numbers of bytes of application data. Approximately ten percent of the received data where actual payload data. One reason for this relatively low figure is the fact that this connection is used for all messages sent from the relay/ rendezvous peer, including all protocol data. The communication actually follows a predetermined pattern. This pattern consists of resolver response messages with the advertisement of the discovered services and data. This pattern was at least obvious in the setup used in the experiment. In addition, different protocol data like the peer advertisement of the relay/ rendezvous peer and resolver queries are injected into the data stream at different time intervals. To make matters even worse, it is not only the advertisements of services located on others peers that are transmitted, but also local services that are registered with the rendezvous peer and match the query sent by the local peer. With the size of one advertisement in mind (see section 5.3.1) the amount of protocol data becomes considerable. The rendezvous does not feature any filtering mechanism to avoid this. In fact the only way to avoid this problem in the current implementation of JXTA, is by carefully constructing the queries.

The *Output* column of Table 5.10 features the amount of application data, in bytes, sent from the Iridium enabled demonstrator node. As with the received data, the amount of application data is low compared to the amount of protocol data. This is also due to a massive amount of protocol data exchange. As the last three columns of Table 5.10 show, the total amount of application data is very low. In fact, in average less than ten percent of all data is actual application data. This is due to the massive amount of protocol data exchanged in JXTA. From this, one might conclude that JXTA and the JXTA protocols are very chatty. This fact is not reconcilable with the environment encountered in a disadvantaged grid, where bandwidth is a scarce resource. Thus, the JXTA protocols must become more efficient.

Capture	Input	Output	Total	Bytes in Total	Ratio
1	41134	37714	78848	756563	10.42%
2	45059	41514	86573	871977	9.93%
3	28019	30856	58875	764163	7.70%
4	35167	39655	74822	891738	8.39%
5	56547	62472	119019	963385	12.35%
Average	41185	42442	83627	849565	9.76%

Table 5.10: Payload ratio of JXTA

6 SUMMARY AND CONCLUSION

P2P technology, and perhaps especially JXTA, is a field of distributing computing that is of interest for the future development of a NBD infrastructure. The features provided by P2P systems like discovery, asynchronous messaging, grouping and the possibility of providing redundancy of services is desirable. The perhaps most desirable feature is the ability of ad hoc organisation. This enables participants to cooperate with a minimum of pre-planing. Another feature is the possibility to subscribe to services (information) in order to only have transferred the relevant information, i.e. the use of a publish and subscribe communication model. However, if JXTA is to become a standard for use in NBD and disadvantaged grids, it must be tuned for the condition it will meet. The main focus of this evaluation has thus been to see how JXTA and the underlying transport protocols perform when placed in a disadvantaged grid. In our experiment a distributed picture compilation demonstrator used Iridium satellite communication to create an as realistic environment as possible. The evaluation has been focusing on both the underlying transport protocol, in this case TCP, and the JXTA protocols. The main reason why TCP has been most in focus is the fact that the approach chosen is most suitable for analysis of low-level protocols. For a more detailed analysis of the JXTA protocols, application level logging must be performed. The interested reader is referred to [19, 11, 12].

In the following sections the performance of both the transport and JXTA protocols are summarised. In section 6.3 proposal for further work are outlined and this report ends with some concluding remarks in section 6.4.

6.1 Performance of TCP

Chapter 5.2 presented the results from evaluating the performance of TCP over one example of a disadvantaged grid, in this case satellite communication provided by Iridium. These results are based on the use of the Windows XP implementation of TCP “Reno” without any additional tuning. The first noticeable overhead is the relative high cost in time of setting up and closing a TCP connection. This is much as expected due to the low data-rate and high latency properties of Iridium. The most noticeable protocol overhead of TCP detected during this evaluation came in terms of packages exchanged. Less than 50 percent of all packages sent contained actual application data. This figure is relatively low and introduces a considerable amount of protocol overhead. The upside of this is that the packages that only contain protocol data is small in terms of number of bytes. In fact when looking at the amount of bytes sent, over 90 percent is actual

application data. Thus, the extra data introduced by TCP is small, but can nevertheless tie up both system and network resources. Some of the data overhead is due to the reliable data transfer mechanism deployed in TCP and especially the use of ACK messages. In addition, the effect of the reliable data transfer mechanism have been evaluated in terms of the number of retransmission.

In addition to evaluating the performance of TCP, it has also been important to evaluate how JXTA utilise the underlying transport protocol. Many of the issues mentioned above can be avoided, or at least the effects can be minimised, by careful design and implementation of applications and middleware. In this evaluation the number and duration of TCP connections used by JXTA are used to denote how effective the use of TCP is. During the evaluation it was found that JXTA use a small number of long lasting connections and many rather short-lived connections. Generally it can be argued that long-lived persistent connections are more effective in transporting data due to lower cost of establishing a connection. On the other hand, such a scheme would tie up resources on both ends of the connection. For servers with high load, i.e. many clients, this could impact scalability. This said, JXTA should be able to exploit the established connections better.

The use of TCP is in many respects advantageous. When used with applications that are not time sensitive TCP guarantees delivery of data. However, for real-time applications with requirements for reliable data delivery its usability may be questioned when placed in a disadvantaged grid. In addition to the mechanisms for reliable data transfer, the flow and congestion control mechanisms deployed in TCP may contribute to a less than optimal utilisation of the available bandwidth. These mechanisms of TCP are designed to work in an end-to-end setting. When designing the flow-control mechanisms in TCP two assumptions were made about packet loss, it either occurs in the network (e.g. a congested router) or at the receiver side. And thus end-to-end flow and congestion control were reasonable. In disadvantaged grids on the other hand errors, due to the low bandwidth, congestion are likely to happen at the sender side, e.g. overflow at the buffer of the radio. Detection of this state may require cooperation within the protocol stack, i.e. cross layer optimisation. In addition it should be mentioned that the TCP “Vegas” version has built in features to minimize the possibilty for this happening. Tuning of TCP parameters like buffer size might also contribute at limiting the effects. These possibilities have however not been investigated any further, as they are not within the scope of this evaluation. Another possible disadvantage of using TCP with JXTA is the fact that it is a unicast protocol. In order to utilise the propagate pipe in JXTA, multicast should be possible.

It should be noted as a final comment on TCP that compared to the overhead experienced by the JXTA protocols TCP performed well. As mentioned before, it is equally important to see how the middleware utilise the transport protocol available.

6.2 Performance of JXTA

Chapter 5.3 presented the results from evaluating the performance of the P2P middleware JXTA when placed in a communication environment similar to a disadvantaged grid. In this evaluation we have concentrated on two metrics, namely the effectiveness of the service discovery mechanism and payload of application data. In addition, the effectiveness of the pipe mechanism has also been mentioned without any usable results.

The results of evaluating the network logs clearly shows that the combination of JXTA protocols and the HTTP/TCP transport used in the experiment is not optimal when considering bandwidth utilisation. JXTA protocols are extremely chatty, that is many messages must be exchanged and the ratio of application data payload is small. In fact, as shown in section 5.3.3, this ratio is about 10 percent application data and 90 percent protocol data, which is extreme. Some of this large overhead is due to the use of XML protocols, which produce a massive amount of extra data to be transferred across the network. An alternative to this is to use the binary message representation, which is identified in the JXTA protocol specification [22]. This may make communication less bandwidth consuming and thus more effective. Still, the amount of protocol data to be exchanged is tremendous due to the chatty nature of JXTA. At the time of writing this message format is only available in the JXTA J2ME implementation. This implementation is protocol compatible with the 2.0 version of the JXTA J2SE implementation.

The chatty nature did also play a role when evaluating the effectiveness of service discovery. As shown in section 5.3.1 service discovery takes a considerable amount of time when performed through a low data-rate channel. This is partly due to the amount of messages to be exchanged and the size of these messages. It may also be due to an ineffective implementation of the service discovery mechanisms. This possibility has not been investigated any further since the focus of this evaluation has been at the communication layer.

The JXTA specification [22] states that the protocols are designed to have very low overhead. During this experiment and the analysis done subsequently it has become clear that the current version of the JXTA specification and the JXTA reference implementation do not achieve this. One might conclude that JXTA did not perform well in a disadvantaged grid. Whether this was due to the JXTA protocols, the reference J2SE implementation of these protocols or less then optimal implementation of the demonstrator, is still an open question. In addition, it should be noted that overhead due to JXTA protocols may be necessary to make the middleware work sufficiently. This has not been taken into consideration during this evaluation.

6.3 Proposal for Further Work

What has become clear while performing the experiment and the evaluation of the performance of JXTA done subsequently is that the chatty nature of the JXTA protocols are not suited for use in a disadvantaged grid with properties such as low data-rate and high latency. Currently there are ongoing work within the JXTA community to measure the efficiency of the implementation and protocols, by providing benchmarks for JXTA implementations [19]. This initiative is much welcomed due to the fact that JXTA is still not a mature standard. Much work is still left in areas like protocol and implementation tuning. The work done in this JXTA community project will in the future probably play an important role in developing the standards further. Hence, this project will be followed closely in retrospect of this evaluation. Although this project and its goals are important the general trend within the JXTA community is towards generating new applications, and not towards more efficient protocol solutions. One alternative is the development and use of the binary message format for data exchange defined in the JXTA specification [22] that may also provide considerable gain in performance, and will hence be followed up.

In this evaluation we have also been focusing on the transport layer and how the JXTA middleware utilise the transport protocol. We have seen that the use of TCP have both advantages

and disadvantages. Disadvantages include high cost of setting up connections and overhead due to the number of packages exchanged without application data content. In the future we propose to use Reliable Multicast Transport (RMT) protocols to reduce these problems. By using RMT protocols the same set of data may be delivered to one or more recipients in a reliable and effective fashion. Most RMT protocols use a NACK based scheme for ensuring reliable transport. Compared to an ACK based scheme, as used in TCP, this may be more economic. In addition, many RMT protocols deploy mechanisms for reducing the chance of NACK implosion and local recovery schemes. The problem is that it does not exist any standard RMT protocol at this time. A survey of RMT protocols of interest can be found in [3]. These protocols must be studied in more detail before integration with JXTA. One possible way of integrating these protocols, and especially the P-Mul protocol [17], is described in [2].

6.4 Concluding Remarks

It should be noted that since this evaluation is not based on a known benchmark the results might not be optimal. And there are no results available for comparison. In addition the number of logs captured is to small to make absolute conclusions of the performance of JXTA. It should however be noted that this evaluation is accomplished using a type of real life application, and communication system which may be regarded as a disadvantaged grid. In addition it is worth noting that the tests are performed within an uncontrollable environment, i.e. using Iridium. In addition the demonstrator start-up procedure may have contributed to some uncertainty. Further in depth analysing of the JXTA communication is needed in order to make a recommendation of its usage in a disadvantaged grid.

References

[1] Brookshier D, Govoni D, Krishnan N and Soto J C (2002): *JXTA: Java P2P Programming*. SAMS. ISBN 0-672-32366-4.

[2] Eggen A (2004): Specification of a Socket Interface for a Generic Tactical Protocol Solution. FFI-Rapport 2004/01162, Forsvarets Forskningsinstitutt (Unclassified).

[3] Eggen A and Haakseth R (2004): A Survey of Multicast Transport Protocols for JXTA over Disadvantaged Grids. FFI-Notat 2004/02243, Forsvarets Forskningsinstitutt (Unclassified).

[4] Ethereal Homepage. http://www.ethereal.com.

[5] Gagnes T (2003): Evaluating Peer-to-Peer Technology for Network Based Defence. FFI-Notat 2003/00563, Forsvarets Forskningsinstitutt (Unclassified).

[6] Gagnes T and Rose K (2003): Use of Middleware in a Picture Production Demonstrator. FFI-Notat 2003/00857, Forsvarets Forskningsinstitutt (Unclassified).

[7] Globalstar Homepage. http://www.globalstar.com.

[8] Gradecki J D (2002): *Mastering JXTA*. Wiley Publishing, Inc. ISBN 0-471-25084-8.

[9] Gustavsen R M, Mevassvik O M and Skjeltorp A (2004): Synthetic Environment of Experiement ”Ad hoc Organization of Picture Compilation” - Technical Evaluation Report. FFI-Rapport 2004/02554, Forsvarets Forskningsinstitutt (Unclassified).

[10] Hafnor H and Olafsen R (2004): Eksperimentering med ad hoc organisering av bildeoppbygging i NBF: Evaluering av operativ nytteverdi - Blue Game 2004. FFI-Notat 2004/01885, Forsvarets Forskningsinstitutt (Unclassified).

[11] Halepovic E and Deters R (2003): The Cost of Using JXTA. In *The third IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, 2003*.

[12] Halepovic E and Deters R (2003): JXTA Performance Study. In *PACRIM’03 Conference, Victoria, BC, Canada, 2003*.

[13] Hansen B J, Gagnes T, Rose K, Mevassvik O M, Olafsen R and Bråten K (2004): Teknologidemonstrator for distribuert bildeoppbygning for nettverksbasert forsvar. FFI-Rapport 2004/, Forsvarets Forskningsinstitutt (Unclassified).

[14] Iperf Homepage. http://dast.nlanr.net/Projects/Iperf.

[15] Iridium Homepage. http://www.iridium.com.

[16] Oaks S, Traversat B and Gong L (2002): *JXTA in a nutshell*. O’reilly & Associates, Inc. ISBN 0-596-00236-x.

[17] P-Mul - A Protocol For Reliable Multicast Messaging In Bandwidth Constrained And Delayed Acknowledgement (EMCON) Environments (Version 1.0). Allied Communications Publication 142 (ACP 142).

[18] Project JXTA. http://www.jxta.org/.

[19] Project JXTA Bench Project. http://bench.jxta.org/servlets/ProjectHome.

[20] Project JXTA Java Reference Implementation. http://platform.jxta.org/servlets/ProjectHome.

[21] Project JXTA JXME Project. http://jxme.jxta.org/servlets/ProjectHome.

[22] Project JXTA Protocols Specification. http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.pdf.

[23] Rasmussen R, Gagnes T, Gustavsen R, Hafnor H, Hansen B J, Haakseth R, Mevassvik O M, Olafsen R and Rose K (2004): Exploratry Experiment ”Ad Hoc Organiztion of Picture Compilation” Conducted During Blue Game 2004:Evaluation Report. FFI-Rapport 2004/01940, Forsvarets Forskningsinstitutt (Unclassified).

[24] Wilson B J (2002): *JXTA*. New Riders. ISBN 0-7357-1234-4.

APPENDIX

A ACRONYMS AND ABBREVIATIONS

ACK	Acknowledgement
COP	Common Operational Picture
ERP	Endpoint Routing Protocol
EWMA	Exponential Weighted Moving Average
FFI	Norwegian Defence Research Establishment
GPRS	General Packet Radio System
HNoMS	His Norwegian Majesty's Ship (Norwegian abbreviation: KNM)
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
J2ME	Java 2 Platform, Micro Edition
J2SE	Java 2 Platform, Standard Edition
JXTA	Juxtapose
Kbps	Kilobits per second
LAN	Local Area Network
MPA	Maritime Patrol Aircraft
NAT	Network Address Translation
NATO	North Atlantic Treaty Organisation
NBD	Network Based Defence
NJHQ	National Joint Headquarters
NULAN	NATO Unclassified LAN
P2P	Peer-to-Peer
PBP	Pipe Binding Protocol
PDP	Peer Discovery Protocol
PIP	Peer Information Protocol
PRP	Peer Resolver Protocol
QoS	Quality of Service
RMT	Reliable Mulitcast Transport
RTT	Round-Trip Time
RVP	Rendezvous Protocol
SRDI	Shared Resource Distributed Index
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
XML	eXtensible Markup Language