FFI RAPPORT

REQUIREMENTS ON SUBMARINE COMBAT SYSTEM ARCHITECTURE

MACDONALD, Robert Helseth

FFI/RAPPORT-2001/05922

FFIE/771/132.1

Approved
Kjeller 17 december 2001

John-Mikal Størdal
Director of Research

REQUIREMENTS ON SUBMARINE COMBAT SYSTEM ARCHITECTURE

MACDONALD, Robert Helseth

FFI/RAPPORT-2001/05922

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
N0-2027 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-2001/05922	UNCLASSIFIED	
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFIE/771/132.1	-	

4) TITLE

REQUIREMENTS ON SUBMARINE COMBAT SYSTEM ARCHITECTURE

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

MACDONALD, Robert Helseth

6) DISTRIBUTION STATEMENT

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

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Submarine	a) Undervannsbåt
b) Combat system architecture	b) Kampsystem arkitektur
c) Commercial off-the shelf	c) Hyllevare teknologi
d) Middleware	d) Mellomvare
e) Quality of Service	e) Tjeneste kvalitet

THESAURUS REFERENCE:

8) ABSTRACT

The use of commercial off-the shelf (COTS) technologies in submarine combat systems introduce inherent and accidental risks for serious failure to meet functional requirements. This has been the experience of the last 10-15 years and have been subject to much research in the naval communities. The main problem arise from poor system integration, technology heterogeneity and unsuitable COTS technologies. The problems are mainly associated with the "infrastructure" of the system, including operating systems, communciation protocols and network (LAN) technologies. In general, COTS technologies do not support the time-sensitive and mission-critical aspect of naval combat systems. They are designed for commercial applications and in general for “best effort” approaches which have no notion of “quality of service”. Bottlenecks in the various layers of the COTS technologies makes system performance almost impossible to predict. Such issues can only be detected at system integration level. One approach to a better design can be by specifying the functional requirements in quality of service terms and then implementing appropriate mechanisms at the various levels of the architecture. This is the preferred method currently being researched at FFI and elsewhere. This study makes some reccomendations to appropriate requirements to be able to catch such problems and presents an analysis of them using the quality of service enabled architecture by using COTS middleware such as the CORBA (Common Request Broker Architecture).

9) DATE	AUTHORIZED BY	POSITION
	This page only	
17 december 2001	John-Mikal Størdal	Director of Research

ISBN-82-464-0572-1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

CONTENTS

REQUIREMENTS ON SUBMARINE COMBAT SYSTEM ARCHITECTURE

1 INTRODUCTION

The work presentation in this study is the result of a brief investigation into the use of commerciall off-the shelf (COTS) information technology in combat systems suitable for future submarines. It is an initial attempt into establishing what requirements must be present in order to avoid potential serious integration problems. The methodology, testbed resources and empirical knowledge has been reused from previous projects at FFI, in particular the New Frigate project.

The main aim of this work is to present a set of recomedations to system level (otherwise known as the System Segment Specification – SSS in the Fridjof Nansen class frigate program) level requirements that capture these inherant and accidental issues of risk.

2 PROBLEMS WITH COMMERCIAL TECHNOLOGY

2.1 Problems with COTS

The combat system (CS) of the future submarines will be comprised of a mixture of traditional military technology and COTS (Commercial of The Shelf) information technology.

The overall system architecture can be defined as being a distributed collection of communicating and collaborating sub-systems with a great variety of communication technologies and protocols. These represent a distributed system, which need to support the desires levels of integration across the sub-systems (which we call “horizontal integration”).

We maintain that the presence of a mixture of real-time bounded (of both hard and statistical “soft” type) and non real-time communication requirements will require the system to support QoS (Quality of Service) in every layer of the system software and hardware. The proper support of QoS in every layer *and* the proper integration between them (we call this “vertical integration”) is paramount to be able support QoS at the system level.

Building distributed combat systems is a difficult task considering the lack of advanced network functions in the operating systems and the sheer heterogeneity of the combat system as a whole.

One common and established way to automate the daunting process of integrating applications horizontally across such heterogeneity of technologies with different performance profiles is through the use of object-oriented middleware. This has the overall aim of hiding the heterogeneity to the top application layer. Many such standards exist, such as CORBA, Java and Microsoft DCOM. CORBA is the technology of choice for parts of the combat system of the Fridtjof Nansen class new frigates.

One problem, which has been widely discussed, is that the most common off-the-shelf operating systems, network technologies etc do not support time sensitive data transfer. Therefore the middleware most often do not either. However have examined the available QoS support in CORBA integrated with ATM and general purpose operating systems in order to be able to evaluate this architecture.

The prevailing argument for using COTS is that optimal cost / effectiveness can be obtained using “internet” based operating systems, general computing hardware and network technologies. Also, in the application domain, COTS high-level languages, approved methods and processing paradigms (such as object-oriented or agent / component based models) this thinking is also valid.

However, experience show that the suitability of a given COTS technology can be questionable to say the least. In general COTS technologies have been designed for a specific civilian market. Applying such technologies into a military mission-critical distributed architecture is non-trivial.

Proper attention must be given to this concern and appropriate measures must be taken in the shaping of the requirements before any acquisition can be made. Safeguards for our specific functional requirements must be generated, and proper analysis and modelling of any tender must be performed.

This study will show that the most critical issue is the implementation of QoS ability at system level as a means of supporting the required real-time requirements. Such abilities govern the overall real- time response and exchange of information between nodes in the system is vital. Failure in this respect will result in systems that can be unstable over time, fail to meet operational requirements (for instance in time-critical modes such as torpedo self-defence) due to high latency or jitter and become unable to fully exploit the potential of expensive resources (such as radar, sonar or sensor systems) due to the bottleneck created in the distributed communication system.

The ability to guarantee application end-to-end QoS relates to the ability at all levels of the system (application domain, operating system, middleware system and hardware resource and network domain) to simultaneously enforce such capabilities. Failure at one level will invariably result in failure of QoS at the system level. This has been the principal shortfall of most of the architectures that have been examined during this study.

System level QoS can be measured in the following dimensions:

- application type (synchronous, "burstyness", time-critical, mission-critical requirements)
- information type (audio, video, hard real-time, statistical or "soft" real-time bounded communications)
- application and user specific requirements (battle resilience, support of legacy resources)

The "system" infrastructure (comprised of the operating system, middleware and network technologies) should form a single distributed system and provide QoS for each and every information exchange. The future trend is towards "adaptive" capabilities where such exchanges are negotiated according to the connection's QoS requirements and provided dynamically by the system.

Finally, COTS technologies have several non-functional concerns intrinsic to their use. The lifespan of a given COTS product follows the cycles and dynamics of the civilian information technology markets. These are rapid and very difficult to track with any degree of accuracy. Typical life spans are from 18-24 months to a few years at best. Spare parts become very difficult or even impossible to obtain shortly after that. Even the standards that these products may be designed according to evolve.

Writing requirements for and ensuring a lifetime support environment for systems made from such components is a major concern. The principal motivation behind the introduction of COTS into military mission-critical distributed systems was to save cost. Yet if this is to be achieved over the expected lifespan of the system (often more than 15 years) and not actually increase cost compared to the traditional military specific technologies, draconian measures in acquisition, maintenance, update and generating specifications must be taken.

2.2 Combat system special requirements

The following table shows what types of data and information are exchanged in a typical combat system.

Main category	QoS parameters	Stacks	Example
Tactical information	Small packages Real-time bound Narrow bandwidth	Middleware	Track databases
High-volume information	Upper time-limit Soft real-time High bandwidth, main burst	Middleware server/client database	Navigation chart data Classification data Sensor data
Non-realtime	Desirable with upper time limit Should extract every possible available resource left at any one moment	Internet protocols Server services	Encyclopedic look-up FTP functionality, user initiated NFS file store
Asynchronous service tasks	Routine or action related service functions No real-time requirements associated but operation should not inflict performance loss on other types of traffic	Internet protocols Middleware	Log copying Backup

Table 2.1. Properties of data exchange message types in combat systems.

2.2.1 The “Best effort” methodology

This is a simple method often preferred by the manufacturers of commercial non-time sensitive products. It has the critical fault that it makes no assumptions on critical issues of time-sensitivity. Basically, it is a simple analysis of each requirement in terms of how much bandwidth (measured in bit/s) each connection across the system will require. This is numerically compared against the (often theoretical) capacity of the interconnect it is to be implemented. As long as there is “sufficient margin” the assumption is that the system will comply with all requirements.

The main fault in this methodology is that the a system designed according to this methodology do not have any mechanisms to differentiate between relative priorities of the traffic that has to share common hardware, such as computer resources and network (LAN).

The “Best Effort” approach do not take into consideration problems of time-sensitivity in the communication between sub-systems or COTS technology bottlenecks in the system. These

are due to non-linear and undeterministic conflicts reducing available bandwidth as all resources are shared equally.

Problems in the true situation is often not possible to detect until a late system integration test. Any approval of the design by the customer must therefore be made conditional to successful system integration tests and exhaustive data load analysis (made to the system level, not isolated to each sub-system) to critical design review, with a draft showing the scope to principal design review.

It is very difficult if not impractical to correct the system if the prior analysis is wrong. This will require system redesign. It may work if the data load is appropriate to the system hardware and software.

A practical demonstration of the entire system is very often impractical as such a facility will not exist at critical design review. If the customer then has “agreed” to the concept of the system integrator changes later may be costly.

This approach is used mainly in older systems where the true real-time and high-volume (such as video) connections are isolated from the tactical data traffic to make the system workable. Thus, the level of integration is low.

Most systems designed according to this principle for the customer has experienced problems which are seldom caused directly to the original analysis but rather to later additions and problems with some sub-systems interaction. Thus the responsibility is very difficult to place on the original system integrator since there is no direct functional link between the causes of unstability and the contractual requirements.

2.2.2 Quality-of-service enabled middleware methodology

Will enforce bandwidth and latency requirements by design. This is done by implementing specific features which allocates resources and ensures that each and every communication requirement is handled in a timely fashion.

Such mechanisms, although not replacing true real-time systems enables higher utilisation of existing systems and makes predictions and analysis of system behaviour possible. By enforcing QoS the system integration is made easier as performance prediction is made possible at an early stage identifying any problems.

Appropriate emphasis must be placed on system integration, and how the various QoS mechanisms in each layer connects to each other.

Connections spanning heterogenous technologies (due to sub-systems from different vendors) may have difficulty in connecting each QoS mechanism in each layer. This is like a chain which needs to be unbroken to work.

It is possible to predict how such a system will behave even at high utilisation of the hardware provided an accurate data load can be provided from the requirements. Potential high risk issues can therefore be detected prior to critical design review from paper analysis only.
It is also possible to predict how future upgrades will impact on the systems behaviour.

This is a new approach that is used in modern distributed systems at varying degrees. Complete packages to implement QoS at various levels defined according to data traffic, or service classes are proposed and implemented by most network technology vendors. Industry standards are implemented as well.

	"Best Effort"	**"QoS Enforcement"**
Method	Simple analysis of numercial bandwidth requirements.	Requirement specifies what quality of service it require, in terms of bandwidth, latency tolerated, error rate acceptable and security level. The infrastructure has mechanisms that enforce and guarantee that these are fulfilled.
Critical Issue	Lack of real-time and time sensitivity requirements of sub-systems and issues across sub-systems. Must include end-to-end perspective and include connections across sub-systems. True performance is a function of system load and this is notoriously difficult to predict.	Simple analysis of what quality of service mechanisms are to be used, and how they interact.
Risk Assesment	High, due to integration problems which do not appear until late system integration. Notoriously difficult to predict performance. Very difficult if not impractical to correct after completion as it lacks fundemental mechanisms to adress time sensitiviness and priorities in general.	Low, since the system enforce the application quality of service requirements.
Example	Example includes KDA MSI-3100, KDA MSI-2005 (as used in the frigates). This was the typical approach of older combat system architectures. True real-time traffic was seldom implemented in the same network but required special technologies.	Most new generation combat system architectures implement some form of this methology.

Table 2.2. Differences between the two most used system designs.

3 COMBAT SYSTEM ARCHITECTURE BOTTLENECK ANALYSIS

3.1 Architecture definition

The 7-layer OSI reference architecture will be used to represent a typical COTS distributed system.

Level	*Role*	*Generic technology*	*As used by*
OSI 5,6,7	Operating system and application interface	Solaris 8 Linux	KDA in MSIFC / NF As proposed by NATO ANEP 56
OSI 4,5	Support efficient, scalable, and automated support for interacting of applications across heterogenous networks.	CORBA	KDA in MSIFC / NF
OSI 4,5	Support efficient, scalable, and automated support for interacting of applications across heterogenous networks.	Java / Jini	KDA in consoles / NF
OSI 3	Support and setup connections across a network.	Transport protocols (TCP, UDP) Internet protocols (IP version4)	
OSI 2	Provide network interconnectivity	Ethernet ATM	

Table 3.1. Main layers of the combat system architecture.

3.2 Principal bottlenecks

CORBA is a popular middleware technology providing support for a distributed application (supporting the object-oriented paradigm) and distribution. It has been available commercially since the early 1990s and has been widely proposed as basis for the integrated weapon system in Modernised Hauk (by DCN) and “National World” for the future submarines (by KDA).

Research at FFI and collaborative research has provided a deeper understanding of the performance issues in CORBA based distributed systems. In general many have experienced latency and scalability problems with the earlier CORBA releases, and questions regarding its support for low-latency and QoS have been raised. These questions have given cause for concerns regarding the suitability of the commercial CORBA implementations for use in mission-critical, real-time or delay sensitive and in high-performance systems.

The main aim of the evaluation was principally to validate a set of benchmark issues developed by research partners (at the Applied Research Laboratory, Washington University in USA) of performance bottlenecks in commercial ORB middleware technologies using a

relevant military data load, and secondly to test the list of proposed optimisations using the TAO experimental real-time CORBA compliant middleware.

This approach would validate the current research on commercial distributed application domains for mission-critical distributed applications, and secondly through the TAO middleware provide a foundation for identifying what qualities and requirements had to be placed on the actual middleware that were to be used in the integrated weapon system.

3.3 CORBA Middleware Performance Overhead

Previous research into middleware performance overhead has focused on the throughput performance issue. These were performed by transferring large amounts of data across networks using "flooding techniques". Such tests are not very relevant to determine performance abilities for military time-critical distributed systems as they miss the special requirements these systems have.

Yet they can show fundamental bottlenecks of the system and is a useful tool to gain a quick picture of the systems capabilities. We performed a simple flooding test first and followed this with proper real-time benchmarks exercised with a realistic data load.

3.3.1 Experimental Testbed Set-up

The testbed “TDF” (Technology Demonstrator Facility) was designed to be able to perform experiments on COTS middleware and network technologies in realistic combat system domain applications. Realistic emulations of actual sensors and QoS requirements provide an environment from which empirical knowledge can be accumulated. This is a particular efficient method when considering complex COTS technologies.

The following technologies were provided in the TDF for these tests.

Traffic Generator	COSSIM v0.7 (see text)
Operating Systems	Solaris 2.6, Linux (kernel 2.2.14 and 2.3.99), Windows NT4.0
Middleware Technology	TAO 5.0 (CORBA v2.4 compliant), RT CORBA 1.0 compliant
Network Technology	WUGS-20 2.4Git/s 2 8x8 ATM switches
	3Com 24-port 10/100 Mbit/s Fast Ethernet switch
Network Interfaces (NIC)	3Com 3c905 10/100Mbit Fast Ethernet
	STS Technologies 1.2Gbit/s APIC ATM
	Efficient ENI-155 155Mbit/s (OC-3a) ATM (2Mbytes RAM)
Basic Host/Node Hardware	9x 266MHz Pentium PC, 2x UltraSPARC 2, 1x SGI Indigo 2

Table 3.2. Testbed Technologies.

The COSSIM v0.7 traffic generator has been developed at FFI as part of the MULTE research program. It simulates the application and sensor loads of naval combat systems according to

specific real-time and QoS specifications. It runs on top of the CORBA (v.2.4) middleware, and has so far been implemented only on ACE TAO 5.0 CORBA.

3.3.2 Traffic Generators

Two kinds of traffic generators were used to emulate the application domain :

- standard “flooding models” transferring untyped and richly typed byte stream data between several hosts using CORBA and lower-level mechanisms like BSD sockets,
- specific application end-to-end generator applying a realistic military combat system data load using a specific range of data types and packet sizes, whilst checking each and every transfer for violations of the specific real-time requirements.

The flooding model is the customary way to measure network and system throughput in systems, which have no real-time requirements. They often give misleading performance figures as the real-world behaviour is a function of many aspects not applied. For instance, real-time & time-critical requirement issues and data granularity will influence on what bandwidth the system can actually deliver.

To be able to evaluate any COTS component the second approach must be used. Both methods are available in the COSSIM traffic generator tool generated at FFI as part of the MULTE research program. It detects violations of defined QoS requirements in the systems by monitoring all data traffic.

Standard methodology from collaborative research and from original work has been used as basis for tests in which specific QoS requirements relevant for mission-critical distributed systems have been applied. Violations of QoS have been detected during long and repetitive testing and data that arrives too late are flagged as invalid.

Realistic and “worst case” traffic patterns were applied from application level, and all measurements were done from an application-to-application level.

3.3.2.1 Parameter Settings

Related research on transport protocol performance over ATM has identified the impact on performance of a number of parameters that can be applied.

- *Socket Queue Size*. The sender and receive socket queue windows were set to “small” (1 Kbyte), “standard” (8Kbyte) and “large” (128Kbyte). The size of this window has been shown to significantly impact on the size of the TCP segment window. In turn this has been shown to significantly impact on CORBA-level and TCP-level performance on high-speed networks.
- *TCP “NO DELAY” flag option*. TCP provides an algorithm (Nagel's Algorithm), which prevents small packets to be sent before acknowledgment of a previous send has been received. Small requests are buffered until such an acknowledgement has been

received. Generally, the flag is set for small packages in time-critical situations, but every test in this study was run with and without this setting.

- *Data Package Size*. A wide range of sizes in bytes were selected in flood-fill scenarios, in increments of 1, 10, 100 and 1000 bytes for the ranges of 1 through 999 bytes, 1000 through 10000 bytes and 10000 through 64000 bytes respectively. For the specific QoS tests appropriate data package sizes relevant to the scenario was used. In addition latency was measured on remote operation invocations, which had no parameters.
- *Numbers of Servants*. Increasing the number of servants on the serves increases the overhead associated with de-multiplexing the server has to perform to send the incoming requests to the servants. A range of servants was used to measure latency and overhead in these operations (1, 100, 200, 300, 400 and 500) on the server.

3.3.2.2 QoS Support in the CORBA v2.4 standard

The previous published standards of CORBA up to and including v2.3 lacked proper mechanisms for implementing QoS. Specifically it lacked features that allowed applications to allocate, schedule and control key processor and networking communication resources necessary to avoid resource congestion.

The CORBA v2.4 introduced the Real-time CORBA (RT CORBA) specifications and the Messaging facility, which support many of the lacking features. The RT CORBA specification defines interfaces for managing the processing and network communication resources. The Messaging facility defines asynchronous and QoS frameworks.

These additional features are outlined in the table below.

QoS Feature	Functionality
Processor resources	thread pools, priority mechanisms, inter-process mutex, global scheduling service
Communication resources	protocol properties, explicit bindings

Table 3.3 CORBA v2.4 Support for QoS.

3.3.2.3 QoS Support in the ATM network technology

QoS has a very specific meaning in the ATM world, and is defined by the ATM Forum organisation. This organisation maintains the ATM standards and is an open industry collaborative effort.

ATM was designed from the outset to support telecommunication voice and video services with stringent latency, jitter and bandwidth requirement on a connection point-to-point topology.

The ATM Forum defines QoS at a low level in the architecture stack (corresponding to OSI Layer 2) in the LANE (LAN Emulation) method of combining IP over ATM. Connections can be made to any other node with specific QoS patterns (a pre-defined mode and appropriate

parameters applied. These are enforced in the network by allocating appropriate resources for the required period of time. This requires specific signalling protocols and mechanism for implementation. For this reason ATM is relatively more expensive than Ethernet technologies.

The modes are as follows :

Mode	Intended Use	Requirement	Notes
CBR	Delay/jitter time sensitive	Fixed and continously available amount of bandwidth for the duration of the connection	Specified in Peak Cell Rate (PCR) of the connection
rt-VBR	Varying amounts of bandwidth with strong regulated delay and jitter	Traffic that is bursty in nature. For instance real-time voice and video conferencing.	Specified by the Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), and burstyness by Maximum Burst Size (MBS)
nrt-VBR	Same as for rt-VBR but with no requirement for time sensitiviness	Examples : Non-real time voice and video.	
ABR	Low cell loss, guaranteed min and max bandwidths but no critical delay or jitter time sensitiviness requirements		Specified by minimum Cell Rate (MCR) and Peak Cell Rate (PCR).
UBR	Applications that use the network on a “best effort” basis.	Examples are email and ftp file transfer.	No service guarantees for cell loss, delay or jitter variations.

As can be seen ATM supports stringent QoS requirements (CBR, rt-VBR modes) and “best effort” type (like Ethernet) modes (UBR).

3.3.2.4 QoS support in Ethernet network technology

Ethernet is designed for a “best effort” paradigm on a connectionless basis. This makes Ethernet difficult to use in real-time systems for two reasons (1) every connection will consume all available bandwidth as there is no mechanisms for allocating resources to a specific connection, and (2) it is impossible to avoid network congestion and transient overloads.

In Ethernet QoS has to be implemented at a higher level in the OSI Reference model. One such method is the Differential Services (DiffServ) implementing different services with different priorities. However, it requires that all participating nodes on the system enforce this standard as it has no QoS guarantees for the layers below. It is therefore possible that one unruly node

can consume all available bandwidth and seriously break system real-time bounds by ignoring the DiffServ mechanism. We did not test any such QoS on Ethernet method but this, clearly, should be carried out as future work as this technology will continue to be available in the low-cost end of the market.

3.4 Analysis of ORB end-to-end performance overhead

One study into the system behaviour of CORBA has recently been performed and published by Boeing (3) in collaboration with several other US research institutions. Many in the defence industries have used this study extensively. However, this study has focused on real-time CORBA's for use in essentially standard non-real time application domains and specifically it does not include end-to-end testing and real-time performance testing.

Considering the simple benchmark test performed using flooding techniques (see below) one can clearly see the disappointing levels of performance. The network layer has a high signal bandwidth (1.2Gbit/s with WUGS-20 ATM and 100 Mbit/s using Fast Ethernet) at the network layer but only a fraction of this is apparent to the application layer. It is also dependent on the size of the application information packages. Small packages fare significantly worse than large ones. Typically, for a 64-byte package only 2-5% of the signal network layer bandwidth is available to the application layer.

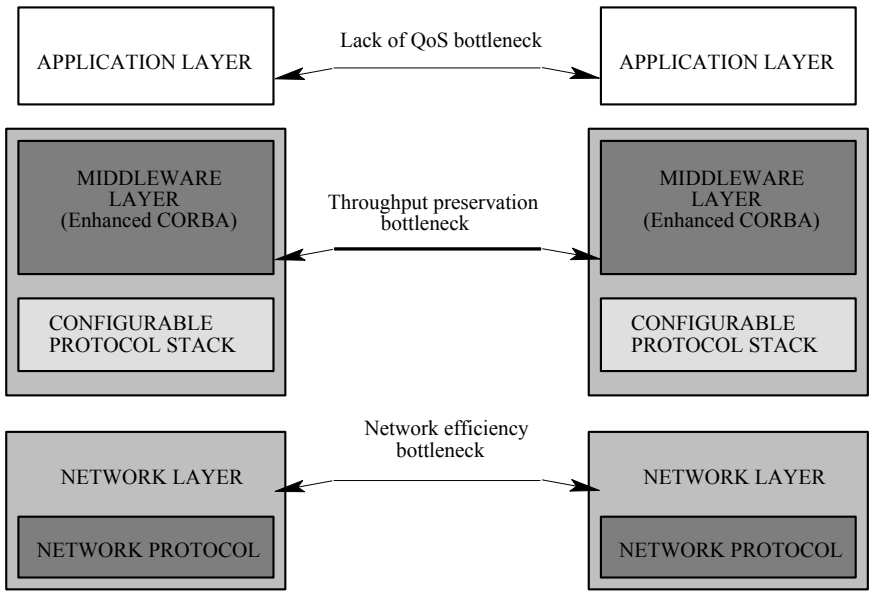

Figure 3.1 Principal bottlenecks in modern distributed systems and principal reasons for performance loss.

The reasons for performance loss is a complex one, and needs to be analysed properly spanning the entire system domain. This must include all the principal layers of the distributed architecture. One principal reason of such performance loss is the general non-determinism in the OS and in application service priority inversion issues resulting from lack of QoS capability in any of the system layers. This is known as the “lack of QoS bottleneck”.

Most research in this field has focused on measuring and optimising the throughput of CORBA ORBs. This is relevant considering the principal commercial application domains and uses of distributed systems in general. It is not relevant for real-time distributed applications and for systems with specific QoS requirements. In such cases it is necessary to also measure latency, latency jitter (variance of latency over time for same repetitive task) and scalability/performance transparency performance.

Experiments were set-up to determine the throughput, latency and scalability performance of an existing CORBA (version 2.4) compliant middleware based architecture with gigabit WUGS-20 ATM network technology. The main operating systems tested were Windows NT4, Solaris 2.6 and Linux (with 2.4.0 kernel).

A set of real-time benchmarks developed for mission-critical systems by the Applied Research Laboratory (ARL) at the Washington University of St Louis, USA, was modified somewhat to reflect our data load patterns. These were compiled and run on the TDF using the ACE TAO (version 5.0) CORBA 2.4 and RT CORBA 1.0 compliant middleware technology, and the ARL WUGS-20 2.4Gigabit/s ATM switch. Comparative measurements were also carried out using a 100Mbit/s Fast Ethernet switch.

3.5 Optimisations for high-performance ORBs

The experimental work shows the limitations of commercial conventional ORB in terms of scalability and latency. The following possible optimisations have been explored to eliminate these bottlenecks in existing ORBs through research collaboration in several research projects at FFI. This work has been based on the open source ACE TAO (v5) CORBA v2.4 compliant middleware and with traffic generators relevant to military distributed combat systems. This work has been carried out at FFI.

A survey of the relevant research obtained and documented in the MULTE project (original published research work, collaborative research and co-operation with several international research institutions) has identified the following areas of principal concern.

- Lack of integration with advanced OS and network features. Existing ORBs do not fully utilise advanced OS and network features.
- Non-optimal de-multiplexing strategies. Existing ORBs utilize inefficient and inflexible de-multiplexing strategies based on layered de-multiplexing.

- Excessive data copying and intra-ORB calls. Existing ORBs are not optimised to reduce the overhead of data copies. In addition these ORBs suffer from excessive intra-ORB functional call overhead.
- Inefficient presentation layer conversions.
- Non-optimised buffering algorithms used for network reads and writes.

3.5.1 CORBA High-performance issues

Based on collaborate research and on external research the following issue have been proposed as being key to latency in high-performance CORBA distributed applications. This issue became focus of special attention and is therefore documented here in more detail.

3.5.1.1 Operation Invocation Strategies

The way CORBA ORBs employs the OIS (operation invocation strategy) may be a cause for increased latency. The OIS determines whether requests are invoked by dynamic or static interfaces and if the client requests a reply or response from the server.

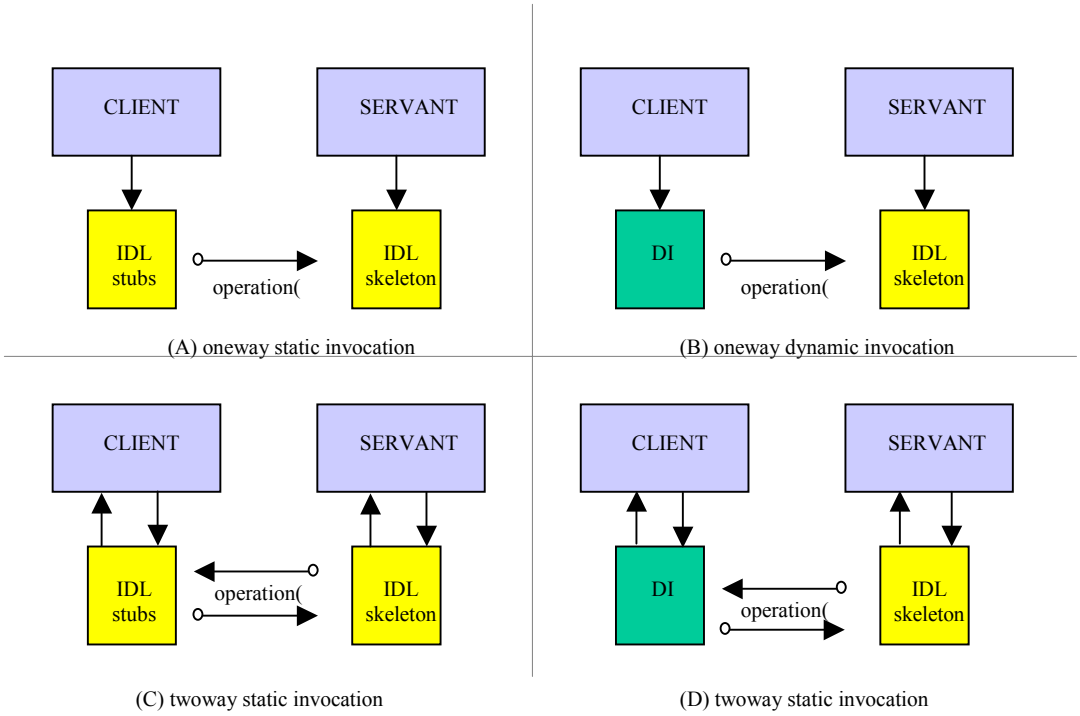

Figure 3.2Invocation strategies tested.

The following OIS were measured and all are based on the CORBA specification.

- One-way static invocation. The client uses the SII (static invocation interface) stubs generated by the OMG IDL compiler.
- One-way dynamic invocation. The DII builds a request at run-time and uses the CORBA request class.

- Two-way static invocation. The client uses the SII stubs for two-way operations defined in the IDL interfaces.
- Two-way dynamic invocation. The DII builds the request and blocks until the call returns from the server.

Both average latency for 100 clients requests were measured and the latency jitter were calculated.

3.5.2 Evaluation Results

See appendix A for a complete overview of results. Only the highlights will be discussed in this section.

3.5.3 Simple COSSIM Flooding Test

Tests were applied across the distributed system with different packet sizes, socket windows sizes and with the NO-DELAY flag of the TCP protocol turned on and off.

The results drawn from this test were as follows. See figure 3.3 for a simplified summary of the measurement. See appendix A for a full overview over measurements. The tests included free flooding throughput, latency, and variation in latency (called jitter).

- The available throughput is dependant on packet size and the available memory allocated to the socket window. Note the theoretical maximum throughput of TCP in figure 4.3 compared to the observed values.
- The effect of turning off the NO-DELAY flag (Nagels Algorithm) was noticable and predictable. Since small packages are less efficient across a network, the Nagels algorithm buffers these at the sender side until a large packet can be transmitted. This increase throughput but an undeterministic wait as a consequence.

The main conclusion is that the network technology do not “deliver” its theoretical peak performance for smaller packets. This must be borne in mind when evaluating theoretical peak performance of technologies such as Ethernet, Fast Ethernet and 155Mbit/s ATM. Note that we used Classical IP (CLIP) as the IP layer for ATM.

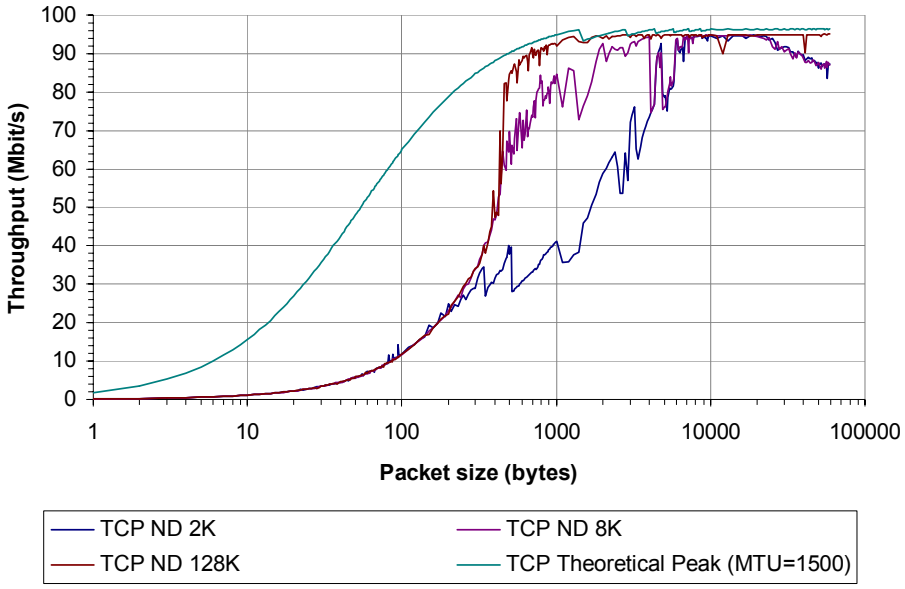

Figure 3.3 *Summary of free flooding results on the throughput test on 100Mbit/s Fast Ethernet. See appendix A for detailed measurements. The TCP (Transport Communication Protocol) and IP was used with varying numbers of socket windows ("small" at 2K, "medium" at 8K and "large" at 128K respectively). NO-DELAY flag is set.*

3.5.4 Real-time COSSIM test with realistic data load

The real-time tests, and optimisation tests are documented in Appendix A. Basically, these are the validation basis for our conclusions below.

These tests show that existing commercial CORBA v2.4 compatible middleware even with QoS capable network solutions (ATM) do not yet have the required level of real-time support for distributed applications in those cases where many objects are involved. They may work in small configurations, but scalability is poor and violates the real-time requirements with even medium sized systems. Employing the RT CORBA 1.0 standard and suitable QoS capabilities in the operating system and network technology did provide significantly better latency and jitter results for our data loads.

Three principal conclusions can be drawn from these tests.

- This validates the conclusion of our research partners for mission-critical military distributed architectures. It does however highlight the fact that the issues of real-time performance are worse for our case.

- The test of the TAO middleware based optimisations do however, show that system level QoS requirements can be met with existing COTS technology if certain optimisations are employed. The problems were most serious in the real-time domain

where the margins for capacities and available bandwidths were small. This implies that proper attention to system integration analysis of QoS and real-time performance must be performed to ensure success.

- The tests also show that without optimisations, or with unsuitable choices of commercial COTS components (operating systems, middleware, network technologies), or inappropriate vertical integration between them, systems will not perform according to intentions. There are qualities bound to principal internals of the system (like for instance the process / thread scheduler or inefficient data copying internally) and would require replacement rather than later stage modifications.

3.6 Discussion of results

See Appendix A for actual plots of the results. As can be seen in figure 3.3the throughput of data measured at the application level varies according to the size of the message being sent from one application to another through the various system layers. Messages below 1000 bytes in size perform significantly poorer than bigger messages. This is due to inefficiencies of the various underlying system layers the message has to transverse.

As most messages in a combat system are between 16 and 64 bytes this is a serious concern. In figures A1.1 and A1.2 multiple messages are exchanged simultaneously. Turning off the Nagel's algorithm in the socket layer will improve throughput (as short messages are accumulated into a bigger one before being sent toimprove efficiency) but this will leave the message in a system message queue until sufficient small messages have been accumulated.

This may be unaccceptable in many time sensitive situations but the effect off turning this mechanism off (in effect trading less throughput for better latency) is shown in figure A1.4. Note that for small messages below 256 byte the performance is only a fraction of the theoretical throughput. In no situiation could the network "deliver" the maximum signal level 100 Mbit/s rate.

Note the effect on setting socket buffer sizes in figure A1.3 which show a much better performance than for smaller socket sizes (figure A1.1 and A1.2) if the socket size is set to 128 kbyte.

Similar measurements for ATM networks are shown in A2, and note in particular A2.2. Such a poor performance is typical of "real-life" situations. Only 1 % of the theoretical performance of the network is available to the application.

The corresponding latency (in section A3) and variance in latency (section A4) is also included.

3.7 Recommendations

The results obtained verified the performance under realistic military loads of commercial and open CORBA compliant middleware for distributed systems. The following principal dimensions of behaviour were measured :

- communication performance (bandwidth/capacity, latency, latency jitter)
- priority inversion
- non-determinism in operations and services

The test system was based on previous and collaborative research work through the MULTE research project, adapted to the military data load defined for these tests. The results of these tests differ therefore from the generic tests they are based upon for this reason, yet they do not entirely invalidate them for our specific need.

Based on these experiences and the tests performed specifically for the submarine project at FFI the following three recommendations are made for the performance, behaviour and capabilities of any candidate of (COTS based) time sensitive operating system and middleware solutions.

- The time sensitive operating system must provide functions for low-latency, deterministic context switching / scheduling capabilities. The tests show that high-latency and high jitter can significantly degrade the ORB performance and predictabilities. System calls can incur significant overhead with threaded kernels when performing process and thread context mode switching.

- The time-sensitive operating system and middleware must support QoS specifications and should have support throughput for enforcing such specifications. Real-time applications often specify their requirements for QoS in quantifiable terms (such as required CPU processing, computation time and periods. In addition the OS should have the ability of allowing enforcement of application domain end-to-end QoS. Note that real-time ORBs cannot provide such application end-to-end QoS unless proper networking support exists for QoS, such as the ATM network technology.

- The real-time operating system should support priority inheritance protocols.

3.7.1 Emerging technologies

The traditional goal of middleware is to mask out problems of heterogeneity and distribution for application developers. With the emergence of new application domains like multimedia and real-time / time-critical applications the flexible support of QoS and real-time requirements becomes a major challenge.

Middleware has emerged as a central architectural component in supporting distributed applications and services. The role of the middleware is to present a higher level of programming paradigm for application writers (typically object-oriented or component-based) and to mask out the problems of heterogeneity and distribution. In application domains where the application itself represents a major part of the cost-driving technologies ensuring a long lifespan is essential.

This applies in particular to the defence domain where the application level runs into millions of lines of high-level code. Much of the infrastructure is comprised of COTS technologies (such as operating systems and networks), which have short lifespans. The middleware layer then serves as an abstract layer to support rapid porting of the existing (often obsolescent) application layer over the ever-changing COTS infrastructure.

However the market for middleware is rapidly changing. For example there is increasing demand to apply middleware technologies in a wider variety of application domains including

- real-time and time-critical systems
- embedded systems
- extreme mission-critical and / or fault-tolerant systems
- distributed multimedia systems
- mobile systems

In this context, flexible and extensible QoS and real-time support is one of the major challenges. These two capabilities are important because research have shown that a single fixed middleware solution (of current technologies) will not be able to support the following requirements that are imposed by the new application domains :

- *Dynamic QoS support*. Applications should be able to specify their QoS requirements and to dynamically change them. The middleware should provide the requested QoS and be able to adapt to changes in application requirement, resource availability and dynamic changes in the infrastructure (a physical change to the system through battle damage for instance).
- *Evolution of QoS requirements*. New media types and new applications might introduce new QoS characteristics. In order to support these new requirements, QoS management in the middleware must be extensible.
- *Transparency versus fine-grained control*. Developers of application components and users should be able to define QoS requirements in high-level (application level) terminology as well as in low-level system parameters to directly influence the configuration of the middleware and resource allocation and re-allocation depending on operation mode and availability of resources at any given time.
- *Policy control*. The middleware should enable end-users, application developers and system managers to specify policies for QoS mapping, negotiation, monitoring, adaptation and so on. For example a policy might be used to express that in case the

quality of a connection should be degraded, adaptation is done by reducing the quality of some aspect, but keeping others according to defined reduced-quality levels (for instance reducing the resolution but not the frame update rate of a sensor video connection across a damaged network).

- *Support for seamless system evolution*. The integration of new components in the middleware should not require re-compilation or changes of existing components and middleware entities. This must also be true for components that encapsulate resources, i.e. API's to network and other resource services.

Through research at FFI it has been the experience with flexible protocol configuration that the above requirements cannot be solved only by dynamic (re-)configuration of communication protocols. Additionally, flexibility in establishing and managing bindings is necessary.

SSS Requirement	*What to look for*	*Acceptable*	*Not-Acceptable*	*Verification requirement*
Formal definition of technical terms such as end-to-end quality-of-service (QoS) and its metrics and verification methology. Also define what involves in the term "max load" according to the system functionalities.	Formal definition of the term QoS, and every term used in it. Must have *formal* referances that can be verified and the suppliers of any COTS technology used must document to which degree they are compliant and to which they are not. The definition must be unambigous and measurable, as it forms the legal interpretation of all subsequent requirements in this area.	Any formal definition as used by COTS manufacturers as long as they state to which degree they are compliant. This must be traced through every layer of the architecture and especially with functionally similar but heteregenous technologies. Definition must include how it is to measured, where it is to be measured, how it is to be measured with reference to the OSI 7-layer reference stack. All data flow in external (specified in IDS) and internal data exchange must be specified in QoS terms.	Assumption on behalf RNoN that technical terms without a formal definition is understood by manufacturer. Most technical terms can be defined in one of many ways.	Examination of the definition and acceptance by expert body.
System behaviour	Mechanisms in the infrastructure that enforce desirable behaviour. If this is implemented across heterogenous technologies in layers then it is neccessary to know which enforcement mechanisms are employed, and how they interact to ensure end-to-end (wrt QoS defintion) desirable performance	Irrespective of any other traffic, all functional requirements with QoS requirements related to the correct behaviour (wrt the QoS definition) should be independent on system load.	Systems time response is a function of the total (even non-related) systems load imposed by applications, users and sensors on the system infrastructure (limited by the end-to-end as applicable to QoS definition)	Analysis of architecture prior to design is frozen, including a data load which defines "max load" in terms of appropriate quality of service terms.
Layered referance architecture.	That the end-to-end perspective is layered in an appropriate way to illustrate what QoS mechanisms are implemented at each layer.	The 7-layer ISO referance layer.	Different models for different sub-systems.	By analysis of design.
Redundancy	The database server is replicated. As resources of the infrastructure are removed the remaining resources should be able to reconfigure to restablish QoS on selected data paths according to prioritised functions.	Proper redundancy.	A centralised system in which the absence of proper redundancy will means loss of other functionalities than those purely related to navigation.	By analysis of design.
System integration	That the specification of the data paths end-to-end (wrt QoS definition) is not split into seperate IDS which splinter the responsibility of "end-to-end".	A complete index of data loads on an end-to-end basis (as relevant for the QoS defintiion) with QoS requirements for each and every path.	Seperate index of data loads specified in IDS documents which has at their limits the various sub-systems of various third suppliers, and at a network / hardware level.	Examination of the index of data loads by expert body to PDR and final version to CDR.
Heterogeneity	Multiple functions implemented with different (but functionally similar) technologies.	The system integrator takes the issue of heteregonous technologies seriously for technical complexity reasons and for life cycle cost reasons. Justify the adoption of functionally similar technologies of different types when they are introduced.	Functional similar technologies in the system.	Examination of the system architecture, and any justifications that the system integrator have for selecting multiple functional similar technologies at any level of the archietcture (ref referance architecture) at PDR.

Table 3.5. Main system levelrequirement framework..

4 CONCLUSION

The SSS must have appropriate requirements to catch the inherent and accidental problems as exists with COTS. This study show that there exists bottlenecks at the middleware and network layers making “best effort” approach perilous at best. In addition there must be attention to how the various layers of the architecture are integrated. Loosely connected sub-systems and heterogeneity of functional similar technologies makes integration difficult.

It is the experience of FFI that these issues are often understood by the manufacturers. However they often seek to avoid to have to implement them. The principal problem is that they seemingly do not wish to do proper analysis and design work prior to freezing the system design (or to comit to a complex design) and even at contract signing could not provide the all important “data load” specification of their system. Lack of suitable SSS requirements will not enable us to put sufficient pressure on the evaluation process.

FFI strongly propose the inclusion of appropriate SSS requirements above those purely of functional nature. A proposal is included in table 4.1.

A. APPENDIX

A.1 Throughput Measurements for 100Mbit/s Fast Ethernet

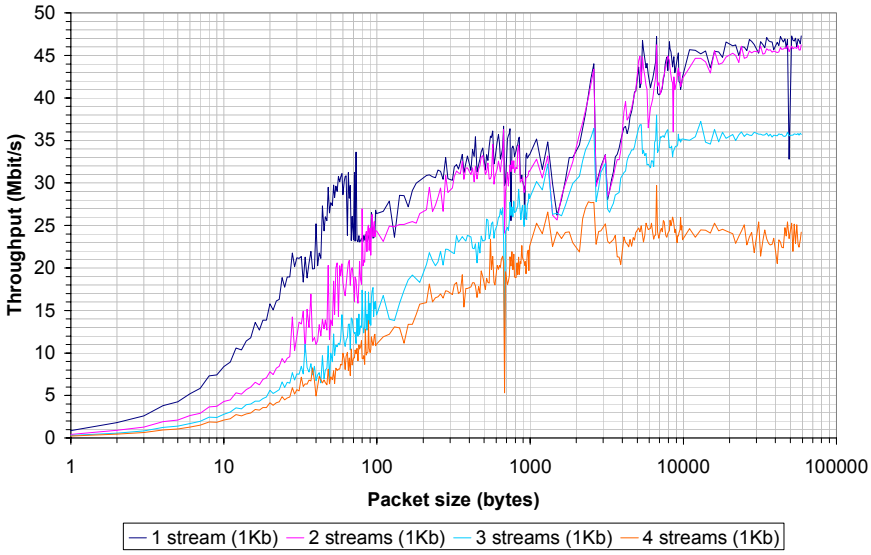

Figure A1.1 Throughput Measurement for 100Mbit/s Fast Ethernet (switched) supporting 1 to 4 simultaneous socket based streams simultaneously. Socket window set to 1K.

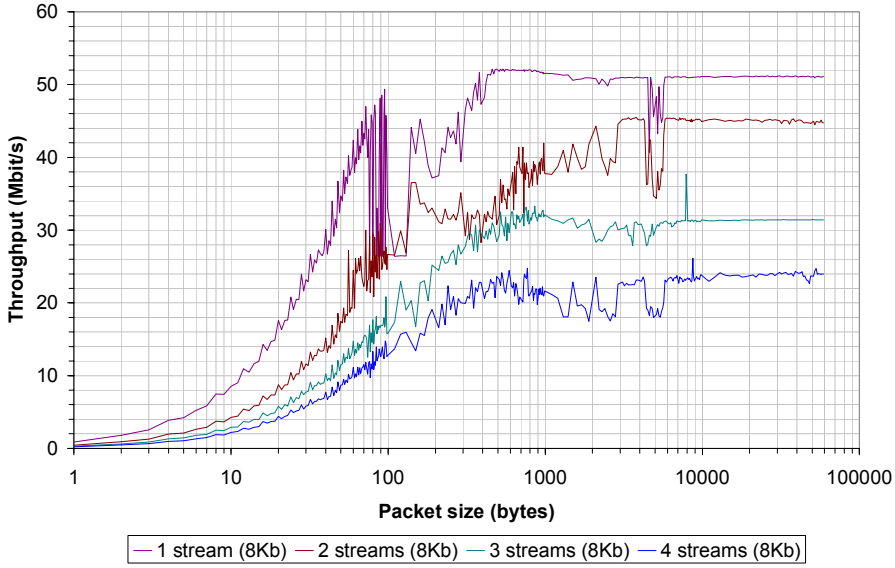

Figure A1.2 Throughput Measurement for 100Mbit/s Fast Ethernet (switched) supporting 1 to 4 simultaneous socket based streams simultaneously. Socket window set to 8K.

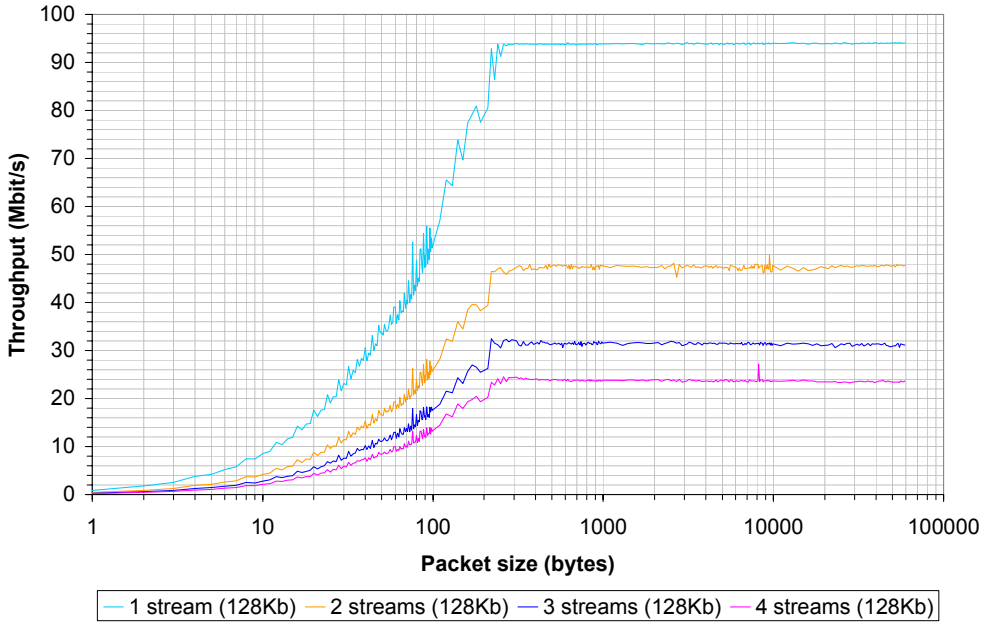

Figure A1.3 Throughput Measurement for 100Mbit/s Fast Ethernet (switched) supporting 1 to 4 simultaneous socket based streams simultaneously. Socket window set to 128K.

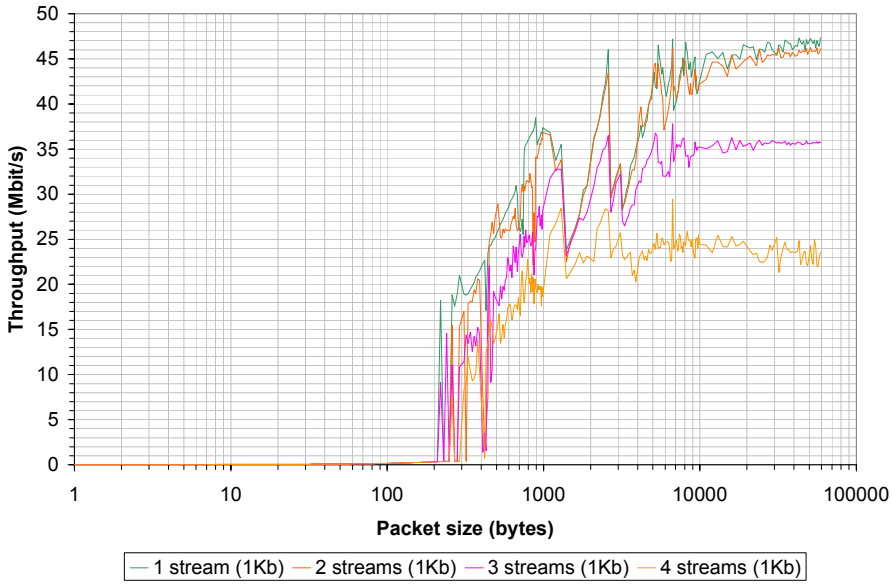

Figure A1.4 Throughput Measurement for 100Mbit/s Fast Ethernet (switched) supporting 1 to 4 simultaneous socket based streams simultaneously. Socket window set to 1K. NO-DELAY flag set to ON.

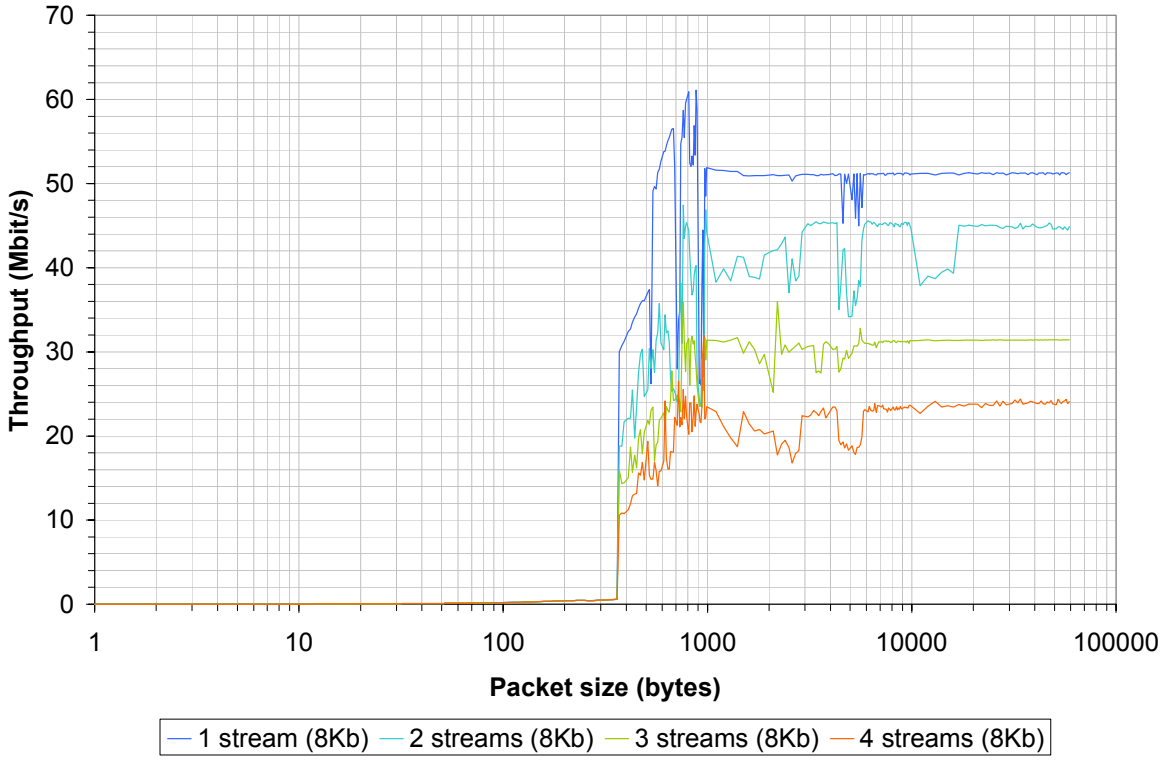

Figure A1.5 Throughput Measurement for 100Mbit/s Fast Ethernet (switched) supporting 1 to 4 simultaneous socket based streams simultaneously. Socket window set to 8K. NO-DELAY flag set to ON.

A.2 Throughput Measurements for 155Mbit/s ATM over CLIP (Classical IP)

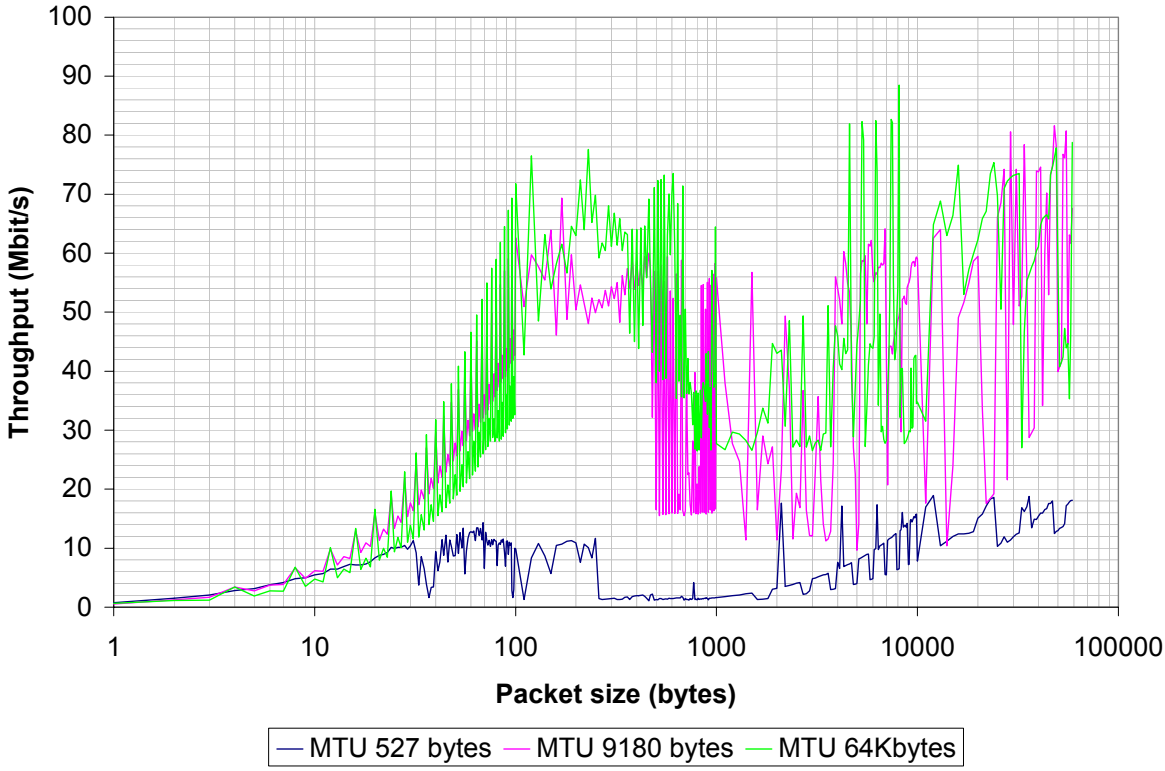

Figure A2.1 Throughput Measurement of 155Mbit/s ATM, and shows effect of varying the ATM MTU value from 527bytes, 8198bytes and maximum value allowed of 64Kbytes. NO-DELAY flag is set to ON.

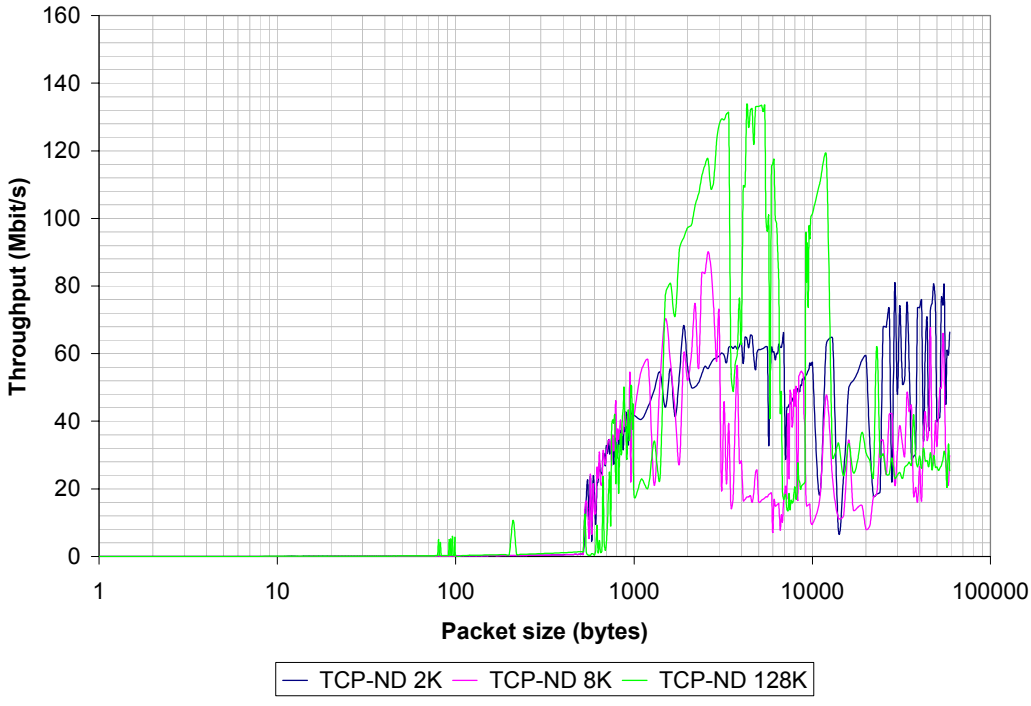

Figure A2.2 Throughput Measurement of 155Mbit/s ATM, NO-DELAY flag set and socket window set to "small" (2K), "medium" (8K) and "large" (128K).

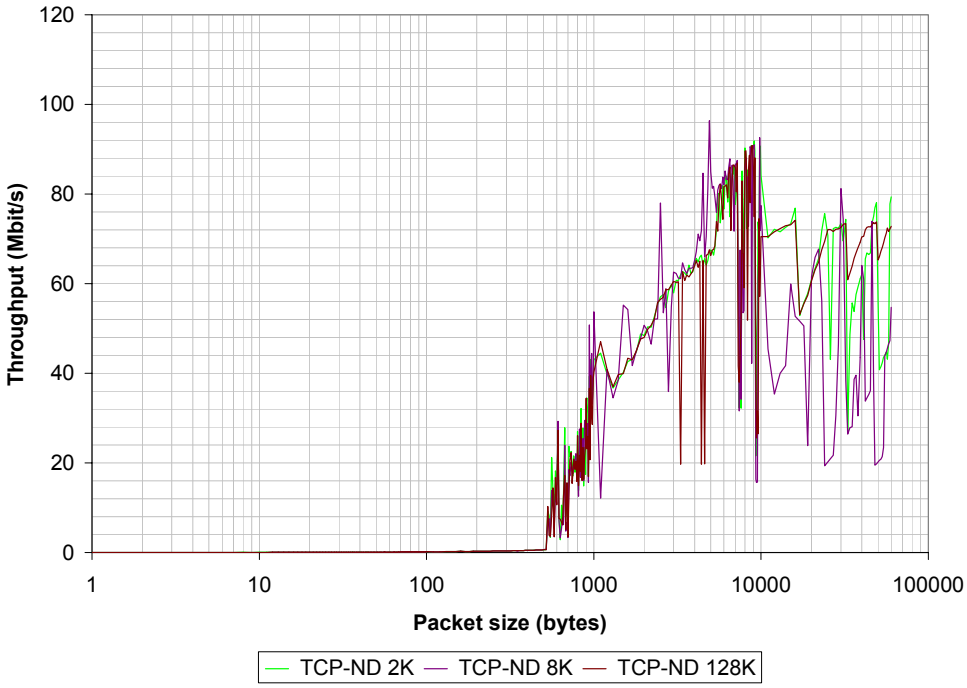

Figure A2.3 Throughput Measurement for 155Mbit/s ATM with NO-DELAY flag set. Socket window set to "small" (2K), "medium" (8K) and "large" (128K).

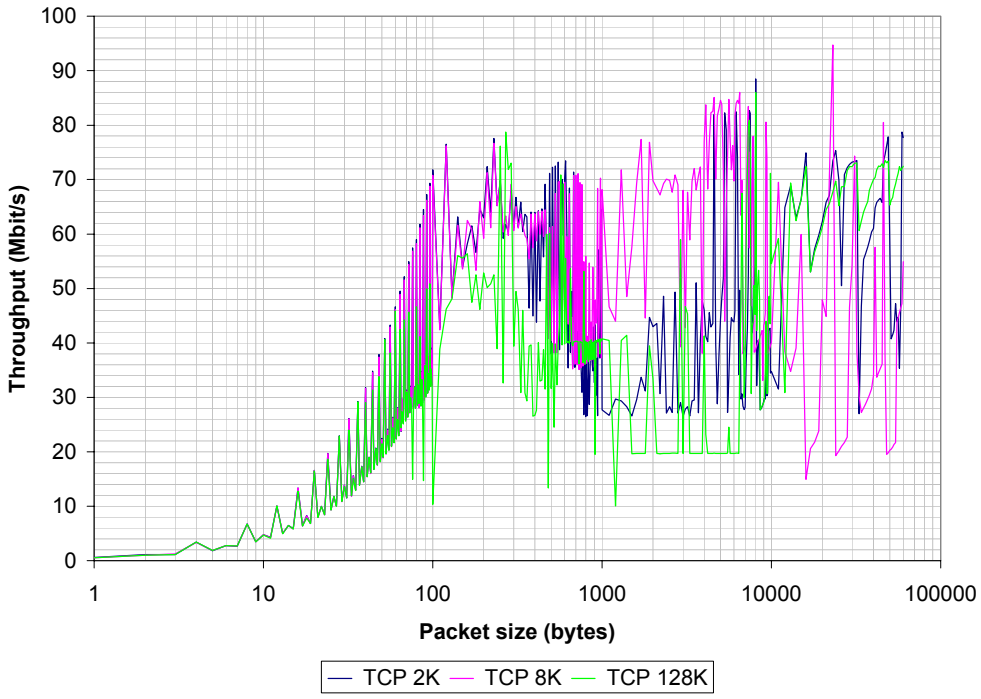

Figure A2.4. Throughput Measurement for 155Mbit/s ATM with NO-DELAY flag set. Socket window set to "small" (2K), "medium" (8K) and "large" (128K).

A.3 Latency Measurements

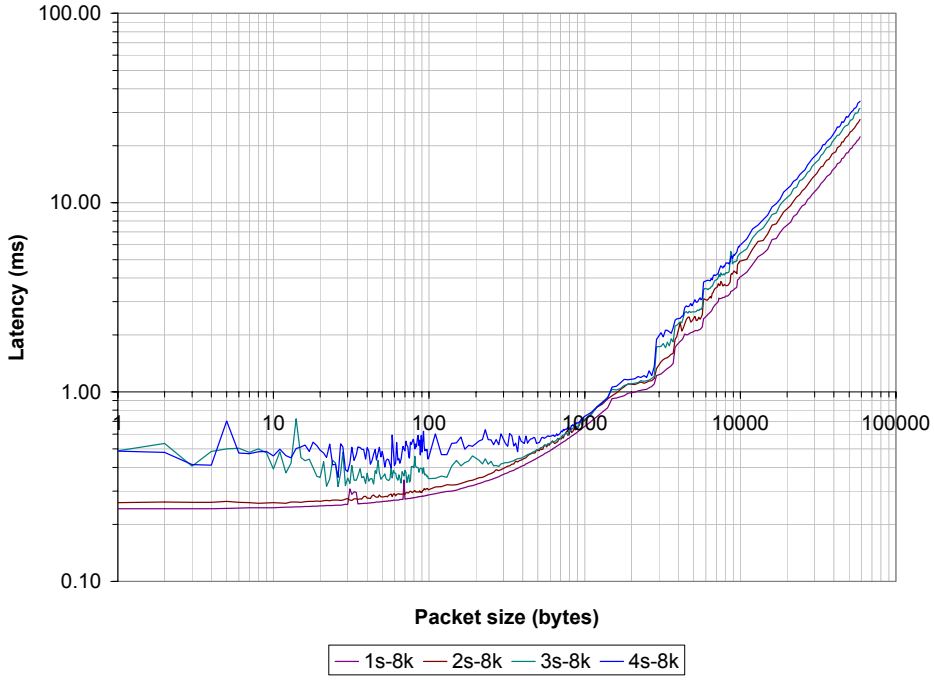

Figure A3.1 Latency Measurement for 100Mbit/s Fast Ethernet, with 1, 2, 3 and 4 simultaneous socket connections running. Socket window is set to 8K.

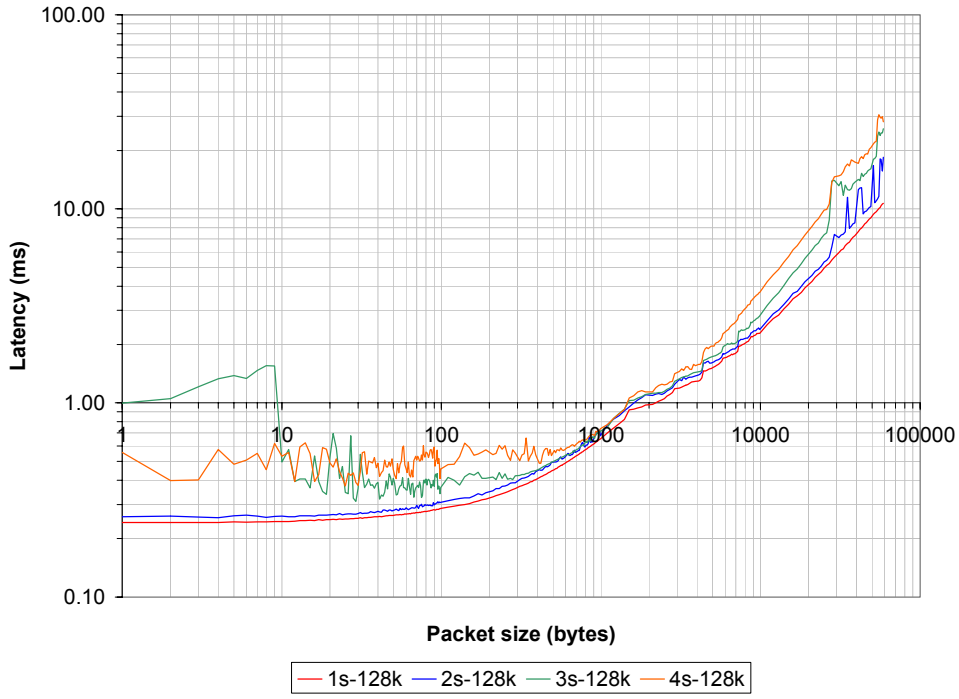

Figure A3.2 Latency Measurement for 100Mbit/s Fast Ethernet, with 1, 2, 3 and 4 simultaneous socket connections running. Socket window is set to 128K.

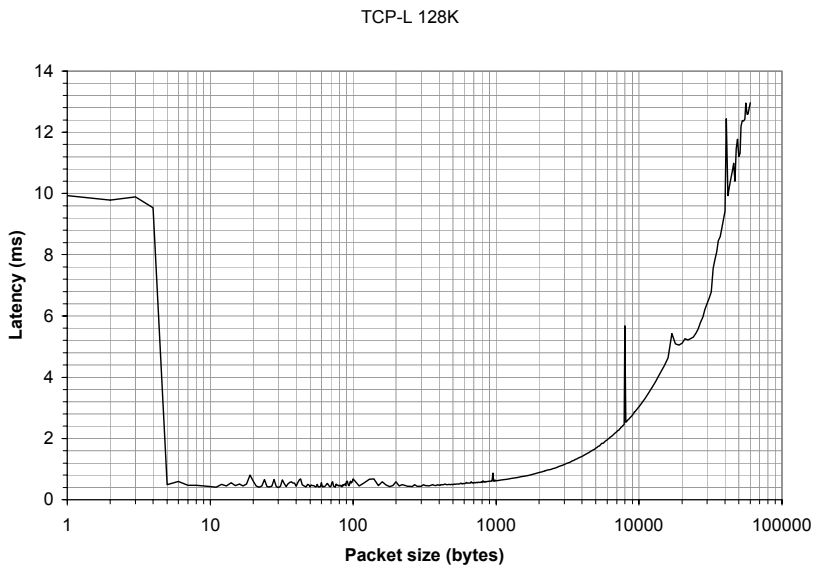

Figure A3.3 Latency Measurement for 155Mbit/s ATM (with CLIP – Classical IP). NO-DELAY flag set to ON and socket window to 128K.

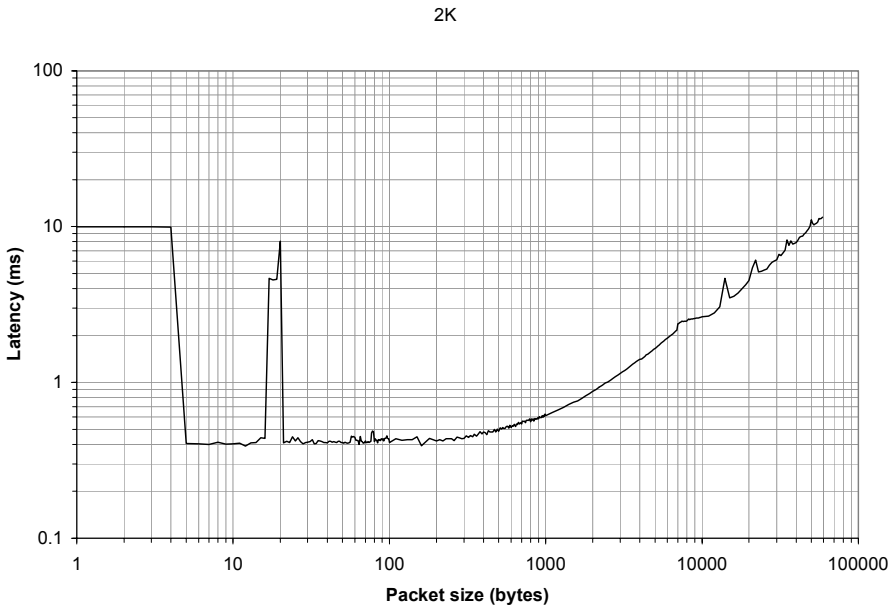

Figure A3.4 Latency Measurement for 155Mbit/s ATM (with CLIP – Classical IP). NO-DELAY flag set to ON and socket window to 2K.

A.4 Jitter Test for 155Mbit/s ATM

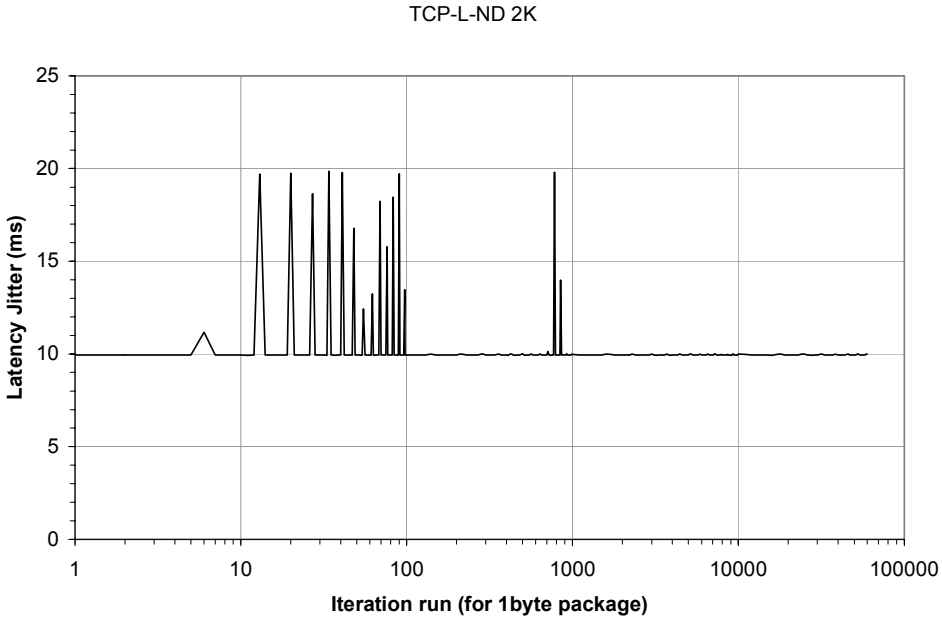

Figure A4.1 Jitter Measurement for 155Mbit/s ATM (using CLIP – Classical IP). NO-DELAY flag set to ON and socket window size is 2K.

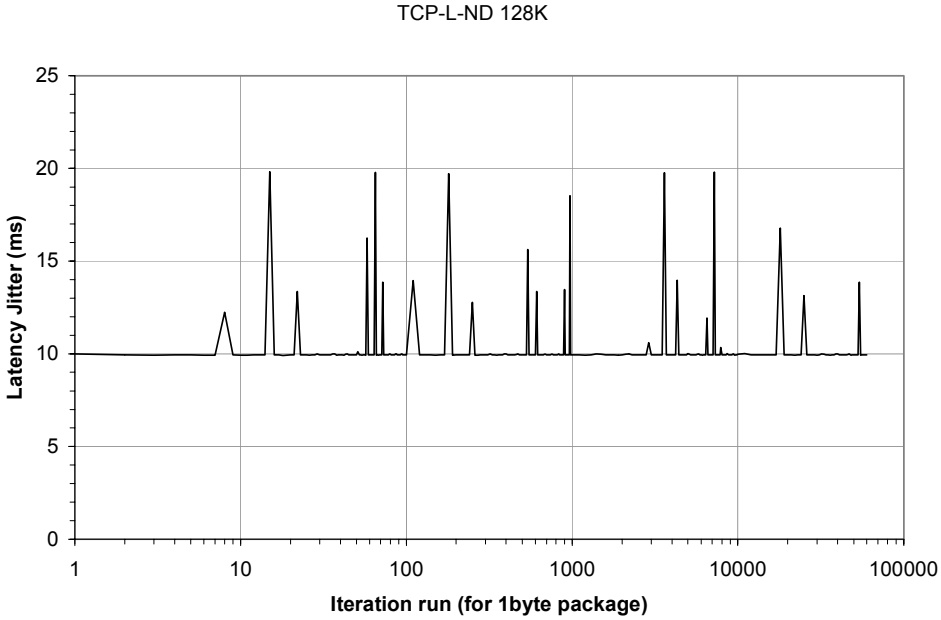

Figure A4.2 Jitter Measurement for 155Mbit/s ATM (using CLIP – Classical IP). NO-DELAY flag set to ON and socket window size is 128K

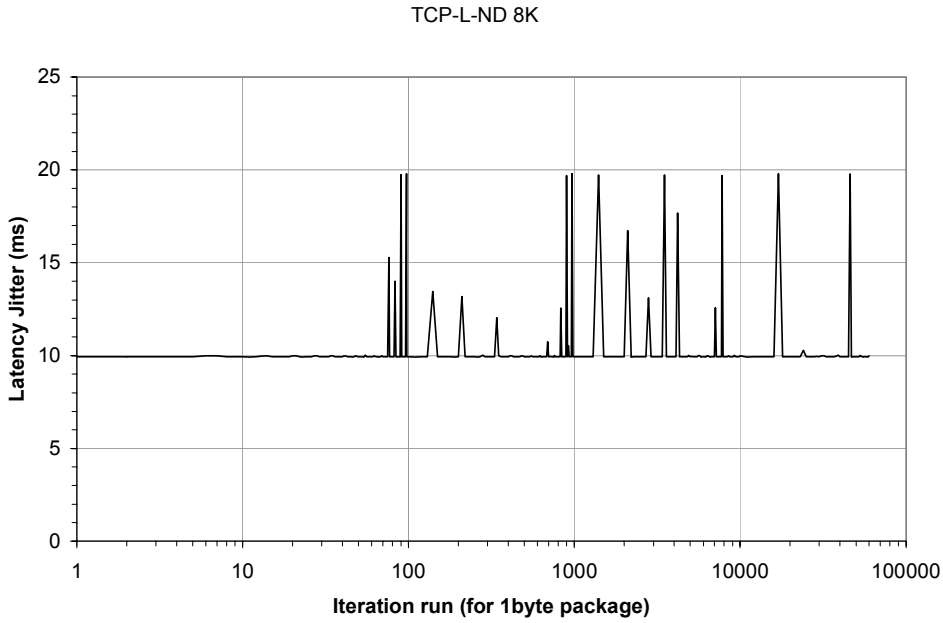

Figure A4.3 Jitter Measurement for 155Mbit/s ATM (using CLIP – Classical IP). NO-DELAY flag set to ON and socket window size is 8K

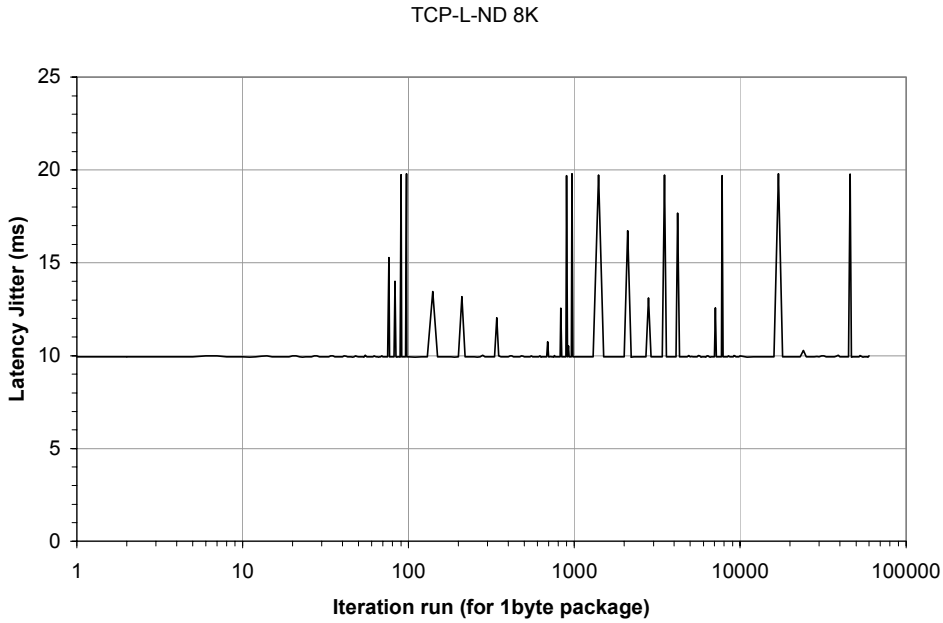

Figure A8.3 Jitter Measurement for 155Mbit/s ATM (using CLIP – Classical IP). NO-DELAY flag set to ON and socket window size is 8K

DISTRIBUTION LIST

FFIE **Dato:** 17 December 2001

RAPPORTTYPE (KRYSS AV)	RAPPORT NR.	REFERANSE	RAPPORTENS DATO
X RAPP ☐ NOTAT ☐ RR	2001/05922	FFIE/771/132.1	17 December 2001
RAPPORTENS BESKYTTELSESGRAD		ANTALL EKS UTSTEDT	ANTALL SIDER
Unclassified		50	39
RAPPORTENS TITTEL		FORFATTER(E)	
REQUIREMENTS ON SUBMARINE COMBAT SYSTEM ARCHITECTURE		MACDONALD, Robert Helseth	
FORDELING GODKJENT AV FORSKNINGSSJEF		FORDELING GODKJENT AV AVDELINGSSJEF:	
John-Mikal Størdal		Johnny Bardal	

EKSTERN FORDELING

ANTALL	EKS NR	TIL
1		SFK/P-PPG03
1		v/OK Tor Arild Orre
1		SFK/T
1		v/Helge Svartveit
1		v/KK Hans Christian Kjelstrup
1		UVBF
1		v/OK Christian Harstad
1		KNM T
1		v/ OK Petter Solheim
1		FO/SST/PLAN-2
1		v/KK Ole Bosse

INTERN FORDELING

ANTALL	EKS NR	TIL
14		FFI-Bibl
1		Adm direktør/stabssjef
1		FFIE
1		FFISYS
1		FFIBM
1		Erik Ahlsen, FFIE
1		Jon Buer, FFIE
1		Karsten Bråthen, FFIE
1		Arne Cato Jenssen, FFIE
1		Stig Lødøen, FFIE
1		Robert Macdonald, FFIE
1		Arvid Melkevik, FFIE
1		Ole Martin Mevassvik, FFIE
1		Kjell Olav Nystuen, FFIE
1		Kjell Rose, FFIE
1		John Mikal Størdal, FFIE
1		Erik Nordø, FFIE
1		Johan Aas, FFIE
1		Pål Kristiansen, FFIE
7		Arkiv, FFIE
1		FFI-veven

FFI-K1 Retningslinjer for fordeling og forsendelse er gitt i Oraklet, Bind I, Bestemmelser om publikasjoner for Forsvarets forskningsinstitutt, pkt 2 og 5. Benytt ny side om nødvendig.