Koffi Patrick N’guessan

Part 1:

ACCEPTANCE TEST AND ADMINISTRATION OF A FARM OF SERVERS

Part 2:

IMPROVING TCP PERFORMANCE IN UNDERWATER WIRELESS SENSOR NETWORKS

2017

Koffi Patrick N’guessan

Part 1:

ACCEPTANCE TEST AND ADMINISTRATION OF A FARM OF SERVERS

Part 2:

IMPROVING TCP PERFORMANCE IN UNDERWATER WIRELESS SENSOR NETWORKS

Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de:

Professor Doutor Amine Berqia

UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

2017

IMPROVING TCP PERFORMANCE IN UNDERWATER WIRELESS SENSOR NETWORKS

Declaração de autoria de trabalho

Declaro ser o(a) autor(a) deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

Assinatura do candidato:_____

The observation precedes the explanation.

Galileo Galilei

Acknowledgment

I would first like to express my gratitude to ATOS SE as well as to the University of Algarve, the two institutions in which I have been involved for the two previous academic years.

The first institution was the host institution during my internship of 6 months in France. My learning has considerably been improved. That is why I would like to express my sincere gratitude to its leadership; particularly to Mrs. Anne Géron, David Marion, Pierre-Marie Tran, Stephane Blanco and Aissam Ourhou.

The second institution has been my academic host. To all members of the Faculty of Science and Technology, I wish to express my deepest gratitude especially to Mrs. Paula Caboz for her availability, guidance, and advice.

Special thanks go to Prof. Dr. Amine Berqia, who kindly agreed to supervise my thesis and provide me with very insightful guidance.

The final thanks go to my family for their unwavering support.

Preface

This paper is intended to complete my master degree from the Faculty of Science and Technology of the University of Algarve. Indeed, it consists of two parts. The first part reports the six-month internship I attended in Grenoble (France) during the first semester of the academic year 2016-2017. The second one corresponds to the research dissertation that was carried out during the second semester of the academic year 2016-2017.Sections ending with "1" refer to the internship, while those ending with "2" refer to the research dissertation.

Abstract 1

During the last decades, companies and organizations have focused on how to provide to the end-users or clients with web services or applications to make them more closer and involved to the activity. Therefore, many enterprises through their direction of the IT service, propose varieties of applications that allow to the stakeholders to perform what they need. The aim of this report is to present what the application integration job is and to report the missions that I have been able to carry out such as application integration, application qualification, and acceptance tests. This represents in total:

- 19 qualified applications,
- 33 administrated servers

Keywords:

Acceptance tests, Application integration, Application qualification, IT service, Server.

Resumo 1

Ao longo das últimas décadas, as empresas e as organizações concentraram-se na forma de fornecer aos usuários finais ou clientes, serviços Web ou aplicativos para torná-los mais próximos e envolvidos na actividade. Portanto, muitas empresas através da sua direcção do serviço de Tecnólogia da Informação TI, propõem variedades de aplicativos que permitem às partes interessadas realizar o que necessitam. O objectivo deste relatório é apresentar o que é o trabalho de integração de aplicativos e as missões que fui capaz de executar, como a integração de aplicativos, a qualificação de aplicativos e testes de aceitação. Isto representa no total:

- 19 aplicações qualificadas,
- 33 servidores administrados

Palavras-chave:

Testes de aceitação, integração de aplicativos, qualificação de aplicativos, serviço de TI, servidor, serviços Web

Abstract 2

Underwater wireless sensor networks (UWSNs) are becoming popular due to their important role in different applications, such as offshore search and underwater monitoring. However, the data transmission in this underwater environment is impacted by various aspects such as bandwidth usage limitation, surrounding noise and large acoustic propagation delays. Therefore, communication itself is an outstanding challenge.

The well-known traditional transmission control protocol (TCP), one of the most used transport protocol on the internet, is not suitable to enable this technology. Even though TCP variants for the wireless network are not foolproof in an underwater environment, their use could probably be more difficult in such a multi-hop communication system. We have chosen Newreno for our study. This variant is a modern implementation that includes the four congestion control algorithms. These algorithms have proved to be effective when it comes to terrestrial networks which could be a basis for our study. In addition, Newreno is known for its algorithm of recovery of several segments lost within the same sending window.

In this dissertation, we have conducted a general study of UWSN technology and examined methods to improve TCP performance in a multi-hop UWSN.

And then, we propose Underwater-Newreno (U-Newreno) our enhanced version of Newreno to improve TCP performance in UWSN. U-Newreno consists of two major modifications: controlling the maximum size of the congestion window and the adaptation of the round trip time (RTT) timeout. The results of simulations carried out with the Aquasim simulator show improvements of performances in terms of gain of:

- packets delivery
- Retransmission ratio of packets delivery.

Keywords:

U-Newreno, Underwater Wireless Sensor Networks, Transmission Control Protocol, Newreno, Performance, Round Trip Time, Aquasim.

Resumo 2

As redes de sensores sem fio subaquáticos (Underwater Wireless Sensor Networks-UWSN) estão-se a tornar cada vez mais populares devido à sua importância em diferentes aplicações, como a pesquisa offshore e monitoramento subaquático. No entanto, a transmissão de dados neste ambiente subaquático sofre devido a vários factores, como a limitação do uso da largura de banda, o ruído envolvente e grandes atrasos de propagação acústica. Portanto, a comunicação é um desafio problemático.

O familiar transmission control protocol (TCP) tradicional, um dos protocolos de transporte mais utilizados na internet, não é adequado para habilitar esta tecnologia. Mesmo que as variantes TCP para a rede sem fio não sejam infalíveis num ambiente subaquático, o seu uso provavelmente pode ser mais difícil num sistema de comunicação de múltiplos saltos. Nós escolhemos o Newreno para o nosso estudo. Esta variante é uma implementação moderna que inclui os quatro algoritmos de controle de congestionamento. Estes algoritmos demonstraram a sua eficácia em redes terrestres que poderiam ser uma base para o nosso estudo. Além disso, Newreno é conhecido pelo seu algoritmo de recuperação de vários segmentos perdidos dentro da mesma janela de envio.

Nesta dissertação, realizamos um estudo geral da tecnologia UWSN e examinamos métodos para melhorar o desempenho do TCP num UWSN de vários saltos.

E então, propomos a U-Newreno (Underwater-Newreno), a nossa versão melhorada do Newreno para melhorar o desempenho do TCP no UWSN. O U-Newreno consiste em duas modificações principais: controlar o tamanho máximo da janela de congestionamento e a adaptação do tempo limite “Round Trip Time”(RTT). Os resultados das simulações realizadas com o simulador Aquasim mostram melhorias nos desempenhos em termos de ganho de:

• entrega de pacotes

• Taxa de retransmissão da entrega de pacotes.

Palavras-chave:

U-Newreno, UWSN, Transmission Control P, Newreno, Performance, Aquasim.

Index

Index of Figures 1

Index of Tables 1

Abbreviation List 1

E

Enedis-IQ	Enedis-Integration Qualification
EDF	Electricité De France
ERDF	Electricité Réseau Distribution France

I

IA	Industrialization and Accompaniment
IQ	Integration Qualification
ISD	Information Systems Department
IT	Information Technology

U

URL	Uniform Resource Locator

Index of Figures 2

Index of Tables 2

Abbreviation List 2

2	2H-ACK	2 Hop-ACKnowledgement
A	ACK	ACKnowledgement
	ADELIN	Adaptive rELIable traNsport
	AODV	Ad hoc On-demand Distance Vector
	ARQ	Automatic Repeated reQuest
	AUV	Autonomous Underwater Vehicle
C	CDMA	Code-Division Multiple Access
	CHARQ	Cooperative Hybrid ARQ
	ChSAP	Channel SAP
	ClSAP	Cross-layer SAP
	CRC	Cyclic Redundancy Check
	CSMA	Carrier-Sense Multiple Access
	CWND	Congestion WiNDow
D	DARPA	Defense Advanced Research Projects Agency
	DBR	Depth-Based Routing
	DDD	Delay-tolerant Data Dolphin
	DESERT	DEsign, Simulate, Emulate and Realize Test-beds
	DFR	Directional Flooding-Based Routing
	DLC	Data Link Control
	DSDV	Destination-Sequenced Distance Vector
	DSR	Dynamic Source Routing
	DUCS	Distributed Underwater Clustering Scheme
E	ERP2R	Energy-efficient Routing Protocol based on Physical distance and Residual energy
	ESRT	Event to Sink Reliable Transport
F	FBR	Focused Beam Routing
	FDMA	Frequency-Division Multiple Access
	FH-MFSK	Frequency-Hopped M-ary Frequency-Shift-Key
	FRT	Fast and Reliable Transport
	FSK	Frequency-Shift-Key

	FTP	File Transfer Protocol
G	GloMoSim	GLObalMObile information system SIMulator
	GPS	Global Positioning System
	GPSR	Greedy Perimeter Stateless Routing
H	HARQ	Hybrid ARQ
	HbH-ACK	Hop-by-Hop ACKnowledgement
	HH-VBF	Hop-by-Hop Vector-Based Forwarding
I	ICRP	Information-Carrying Routing Protocol
	ISI	Inter-Symbol Interference
	IW	Initial Window
L	L2-ABF	Layer by layer Angle Based Flooding
	LASR	Location-Aware Source Routing
	LCAD	Location-Based Clustering Algorithm for Data Gathering
M	MAC	Medium Access Control
N	NAM	Network AniMator
	NCIA	Network Coding with Implicit Acknowledgment
	NCRF	Network Coding in Rateless Fashion
	NS	Network Simulator
	MIRACLE	Multi-InteRfAce Cross-Layer Extension
O	OLSR	Optimized Link State Routing
	OTCL	Object-oriented extension of Tcl
P	PAR	Positive Acknowledgement and Retransmission
	PARSEC	C-based parallel simulation language
	PSK	Phase-Shift Keying
	PVC	PolyVinyl Chloride
Q	QAM	Quadrature Amplitude Modulation
	QELAR	Q-learning-based Routing
R	REBAR	Reliable and Energy Balanced Routing Algorithm
	RF	Radio Frequency
	ROS	Robot Operating System
	RREQ	Route REQuest

	RREP	Route REPly
	RTS	Reliable Transport and Storage
	RTT	Round Trip Time
	RWND	Receiver WiNDow
S	SACK	Selective ACKnowledgment
	SAP	Service Access Point
	SBR-DLP	Sector-Based Routing with Destination Location Prediction
	SDRT	Segmented Data Reliable Transfer
	SMSS	Sender Maximum Segment Size
	SSTHRESH	Slow-Start THRESHold
T	TCBR	Temporary Cluster-Based Routing
	TCL	Tool Command Language
	TCP	Transmission Control Protocol
	TDMA	Time-Division Multiple Access
	TTL	Time-To-Live
	TWSN	Terrestrial Wireless Sensor network
U	U-Newreno	Underwater-Newreno
	UW-A	Underwater Acoustic
	UWSim	UnderWater Simulator
	UWSN	Underwater Wireless Sensor Network
V	VBF	Vector-Based Forwarding
	VBF-NC	Vector-Based Forwarding Network Coding
W	WOSS	World Ocean Simulation system
	WSN	Wireless Sensor Network
	WWW	WorldWide Web

Part 1: Acceptance Test and Administration of a Farm of servers

1 Introduction

Over the years, the life cycle of companies have constantly been experiencing changes that are due either to strategic managerial reorientations, to the new configuration of the targeted market or even by the adoption of new daily attitudes such as payments via personal mobile devices. To do this, companies that wish to keep competitive, adopt the principle of continuous integration of their business applications in order to get closer to the real world and consequently satisfy their customers.

The principle of application integration is based on the continuous integration. The latter is defined as a set of practices consisting of checking with each modification of source code that the result does not produce regression in the developed application. The advantage of this process is that it does not forget elements during the production-start and therefore to improve the quality of the product.

In this regard, I completed a six (6) month internship with the topic: acceptance tests and administration of a server's farm. This report reflects the progress of my work, and is divided into five (5) sections as follows: Chapter 1 presents the host organization, the client and the actors involved in the mission. This chapter also presents the theme of the internship. Chapter 2 is an overall background of the Enedis-Integration Qualification (Enedis-IQ). It presents a state of the art of the work tools namely document sources, tooling. Chapter 3 is devoted to the realization of the mission. Chapter 4 provides a personal assessment of this internship based on points such as results, professional and interpersonal skills acquired. Chapter 5 concludes the report.

1.1 The host organization

Atos is an international leader in digital transformation with an annual turnover of around 12 billion euros. Atos has 100,000 employees in 72 countries [1].The company delivers technologies that accelerate the development of its customers businesses and help them to realize the vision of the company of the future while offering its employees a range of various technologies. Atos' activity is organized around 4 businesses [2]:

• Consulting,

• Integration of systems,

• Outsourcing,

• Transactional services (Atos Worldgrid).

Atos has evolved since its creation. *Figure 1.1* gives the illustration of the history of Atos

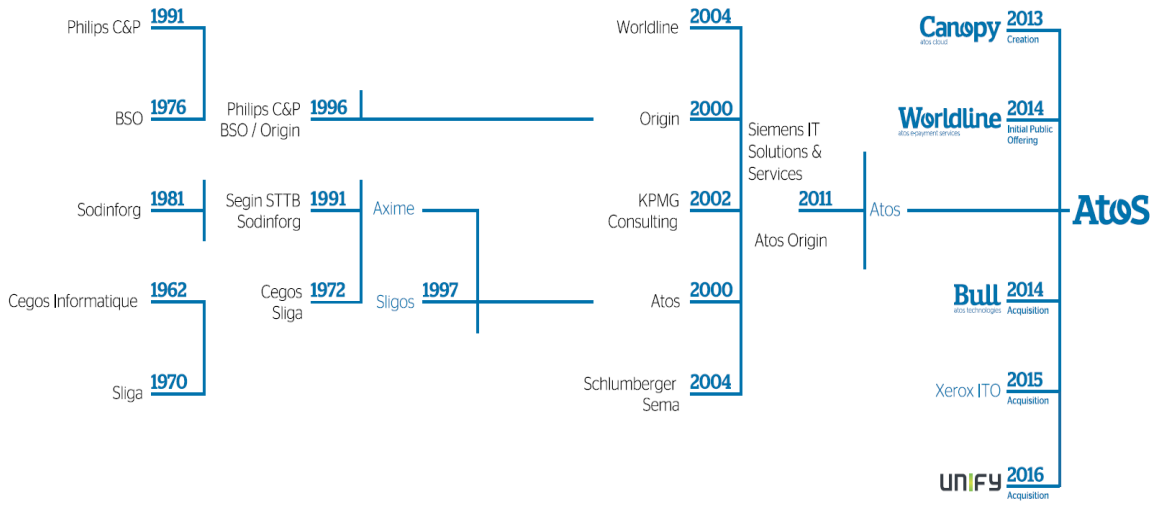


Figure 1.1 Evolution of Atos from its creation until 2016 [3]

The brands of the group are represented by *figure 1.2* below [4]:

Figure 1.2 Brands of the group [4]

Atos offers its services to major international accounts in all sectors of activity. Among these major accounts are [5]: CNAF, UGAP, Auxylium, Lutèce, Informadis, AccorHotels, PMU, CityPulse (Eindhoven), Festival d'Avignon, Ecocité (Grenoble), Météo-France, Keolis, Philharmonie, le CH de Valence, DGAC, Enedis (former ERDF), Omega telecom/ Virgin

Mobile, Pernod Ricard. In [6]it is outlined the technology partners namely Cisco, Dell, EMC, Microsoft, Oracle, SAP, Siemens, VMware (desktop virtualization and infrastructure), HP, and IBM.

1.2 The client: Enedis

Formerly ERDF (Electricité Réseau Distribution France), a subsidiary of EDF (Electricité De France), Enedis is responsible for managing 95% of the electricity distribution network in France. Its mission is to [7]:

• manage public electricity distribution networks in the continental metropolitan area
• operate, maintain and develop these networks
• design, construct works and supervise works
• negotiate, conclude and manage concession contracts
• guarantee users access to networks in objective, transparent and non-discriminatory conditions
• ensure interconnection with other networks
• count consumption by users, providing meters, installing, controlling, maintaining, renewing and managing the data collected
• provide services for local distribution companies and local authority bodies

To enhance its new projects, Enedis has decided to decentralize its Information Systems Department (ISD) to different sites in Paris and Lyon. In order to meet the operational requirements of the infrastructure and applications, a department within ISD was created: Information Technology (IT) operator Enedis. It is an ISD entity whose mission is to create and ensure the implementation of a reliable, efficient, practical, coherent and evolutionary information system. The missions of the IT operator are [8]:

• The exploitation of Enedis project management applications
• Providing technical and functional support for business applications
• Ensuring the office and telephone outsourcing of the national functions of Enedis
• Ensuring operational monitoring of the security of IT and telecom infrastructures

1.3 The actors of the IT operator-Enedis

During a production, the following actors are addressed [9]:

- IA (Industrialization and Accompaniment) accompanies upstream IQ (Integration/Qualification) projects by explaining the processes as well as demonstrating the documents to be delivered as input to IQ. They support the IQ on a technical level and carry out technical approvals.

- PEPSA: Support team on the most important applications at Enedis.
- IS Team of experts on the main technologies used by Enedis. They are in support of the other teams and indicate the standards to be respected within the OI-Enedis.
- Application Manager manages one or more applications, from recipe to production. It is the coordinating link between. It is the main interlocutor of the IQ.
- Enedis-IQ team that ATOS Integration is the provider, is in charge of integrating and qualifying the application
- The production outsourcing team

1.4 The internship topic

This section introduces the topic: Acceptance test and administration of a farm of servers. This theme is therefore divided into two parts:

1.4.1 Acceptance test

It is about the implementation of tests in order to enable the functioning of applications properly. This notion will be fleshed out in section 3.7.3.

1.4.2 Administration of a farm of servers

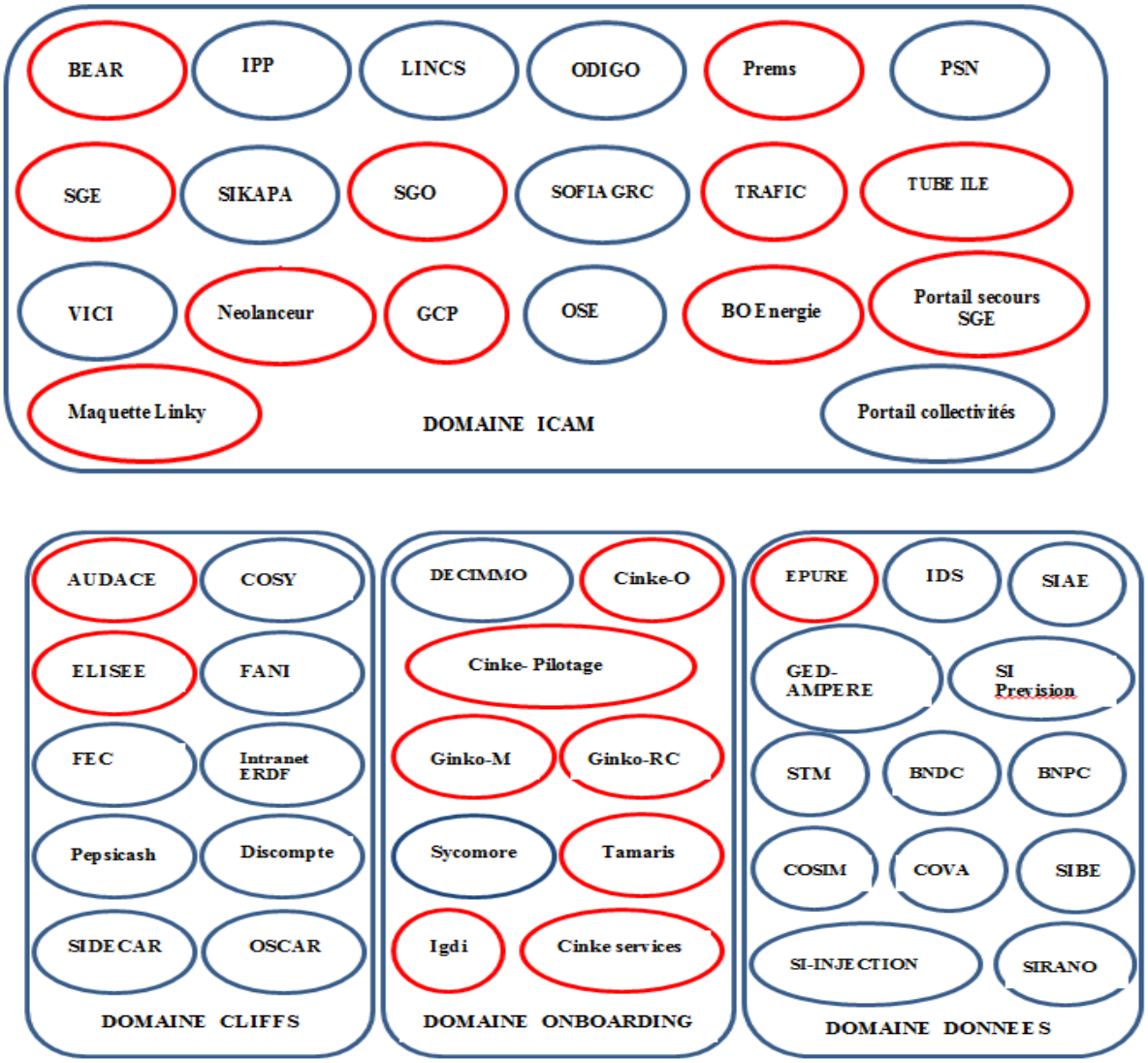
A farm of servers consists of a set of virtual servers on which are implemented the software strains for building the technical infrastructure of the Enedis business applications. Their administration is meant to update and upgrades this architecture and also doing the necessary actions to implement the applications and make them working and available. This administration can be done either by an IQ (Integration and Qualification) or auxiliary by the means of a ‘work request’.

2 Background

My position as a trainee and newcomer to the mission, made me understand that the first objective was to learn and assimilate the context of the mission to carry out each project of the mission in autonomy. This section refers to the study of the state of the art of the Integration/Qualification (IQ). We will discuss the client's business applications; the various support documentation, the tools to optimize the exploitation of applications as well as the implemented platforms that constitute the functional architecture of an application.

2.1 The applications

Below is the list of Enedis applications classified in 7 domains:



2.2 Documentation materials

2.2.1 WikIQ

This is a server available via the Enedis intranet [10] that provides information about all or part of the integration and qualification environment (application and tooling). It works as a search engine dedicated to the IQ, including first-line information such as client applications, system platforms (windows, windows server, Linux, AIX), tooling platforms (Control-M, Patrol, squash), software platforms (Oracle, MySQL, Weblogic, SAP, ...)

2.2.2 Capdoc

It is a web-based collaborative document development application based on the Nuxeo DM enabling: multi-document production, use of pre-filled templates, action tracking (creation, deletion, modification of a document), workflow management,

validation, advanced rights management, automatic generation of the document in PDF format, generation of reports in CSV format, publication of the document in Web format, management of obsolescence. Capdoc is the reference tool for the development of the DEX (or operations file), DAT (or technical architecture file), PCS (or service continuity plan). A description of the procedure of creation of these documents is shown in [11].

2.2.3 The MySI portal

The MySI portal of Enedis is a very stable web platform that includes: IQ module (essential for integration / qualification), the production-start application for an IQ outbound delivery to the outsourcer, the production-start infrastructure and the classic shutters DEX, DAT, PCS. It is available internally via [12].

2.3 Exploitation of an application

The operation of an application is done through technical documents namely:

- DEX includes information such as the general description of the application (identification, history, operating context, general production architecture, products required, implantation, treatments under control-M), the operating instructions of the application (supervision, functional tests, shutdown / restart procedure, availability test, backups, purges, restoration of the application), the security management (accounts to access to servers, products, management of sensitive and confidential data, traceability, logs, etc.)
- DAT is a documentation containing the functional and application architecture, application flows and processing, the issue of application security related to the National Commission for Information Technology and Civil Liberties declaration and the certificates used, general technical architecture choices, technical infrastructure, and the robustness of the solution with respect to availability, the recovery plan, the scheduling and the production process.
- PCS is a document that brings together all the operational information contributing to the activities of managing the availability and continuity of services of the application.

2.4 Tooling

The term "tooling" means the entire hardware and software set up to properly manage the operation and exploitation of business' applications. The most used are: Squash, Patrol, Control-M, Envmep.

2.4.1 Control-M

Control-M (version 8.00) is the tool used to industrialize task (also named job) scheduling. The scheduling is done through a scheduling package provided by the project. It describes in an Excel file the jobs to be executed in order to constitute a chain of an application. *Figure 2.1* below illustrates a control-M chain console

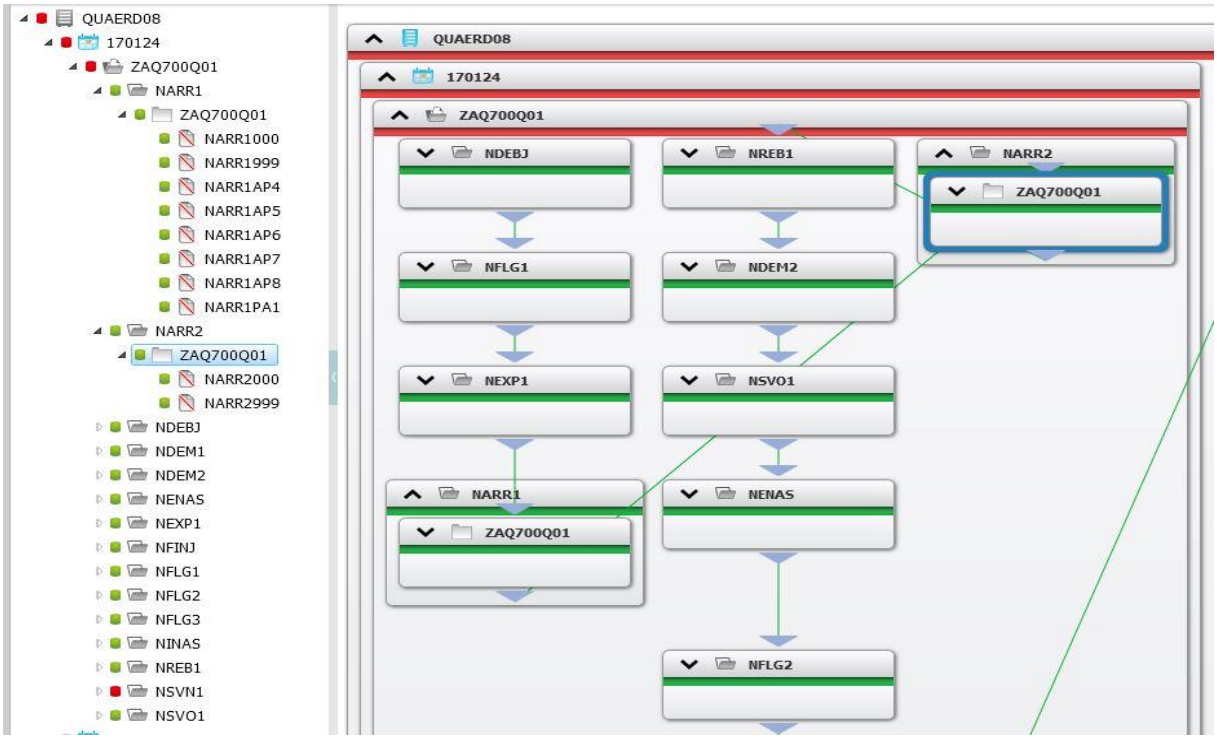


Figure 2.1 Illustration of a control-M chain

Figure 2.1 exhibits information as follows:

QUAERD08 : the name of the host server

170124 : the date of scheduling

ZAQ700Q01 : the name of the application control-M chain (Application: ACTEUR DU SI)

NARR1, NARR2, and NDEBJ: groups containing jobs

NARR1000, NARR2999: refer to different jobs

2.4.2 Patrol

PATROL (version 9.0) is the monitoring tool. Its role is to ensure the stabilization of the system by anticipating problems that may arise. On the Patrol console (*cf. figure 2.2*), the alerts are transmitted in real time with details allowing the integrator to troubleshoot them or to report them to the project team when it is outside

its perimeter. In order to be supervised an application requires the deployment of the specificity of the applicative monitoring).

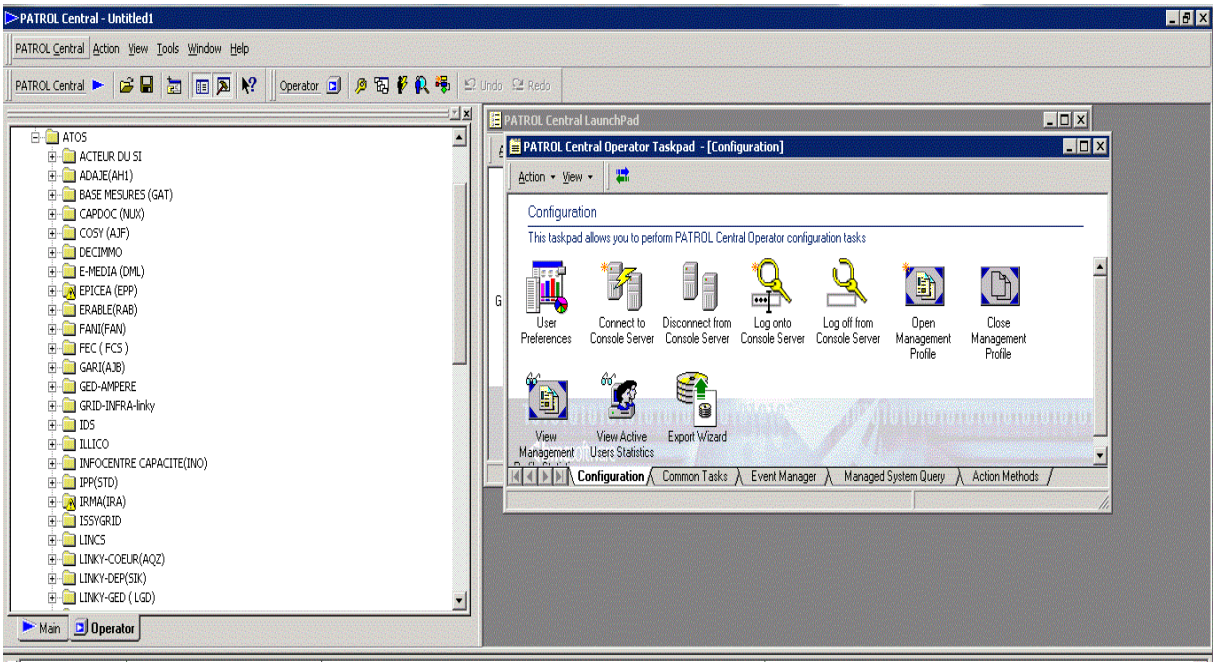


Figure 2.2 Illustration of the Patrol console

2.4.3 EnvMEP

The EnvMEP (or production-start environment) version 2 is a tool for exploiting the production environment. It allows harmonizing on all servers the methods of stop products, start products, test products, rotations of logs, purges of logs, reboot, backup, etc. This tool is an important support for the Control-M scheduler for jobs related to these methods.

2.4.4 Squash

The full integration / qualification process includes an essential part devoted to the tests: prerequisite, installation, verification and acceptance test. To do this, the appropriate tool for these tasks is "Squash" (*cf. figure 2.3*).

Functionally, depending on the type of the desired tests, the integrator constitutes a campaign which is a composition of test cases that can be launched either automatically or manually. An execution report is generated for the status of the tests (*cf. figure 2.4*).

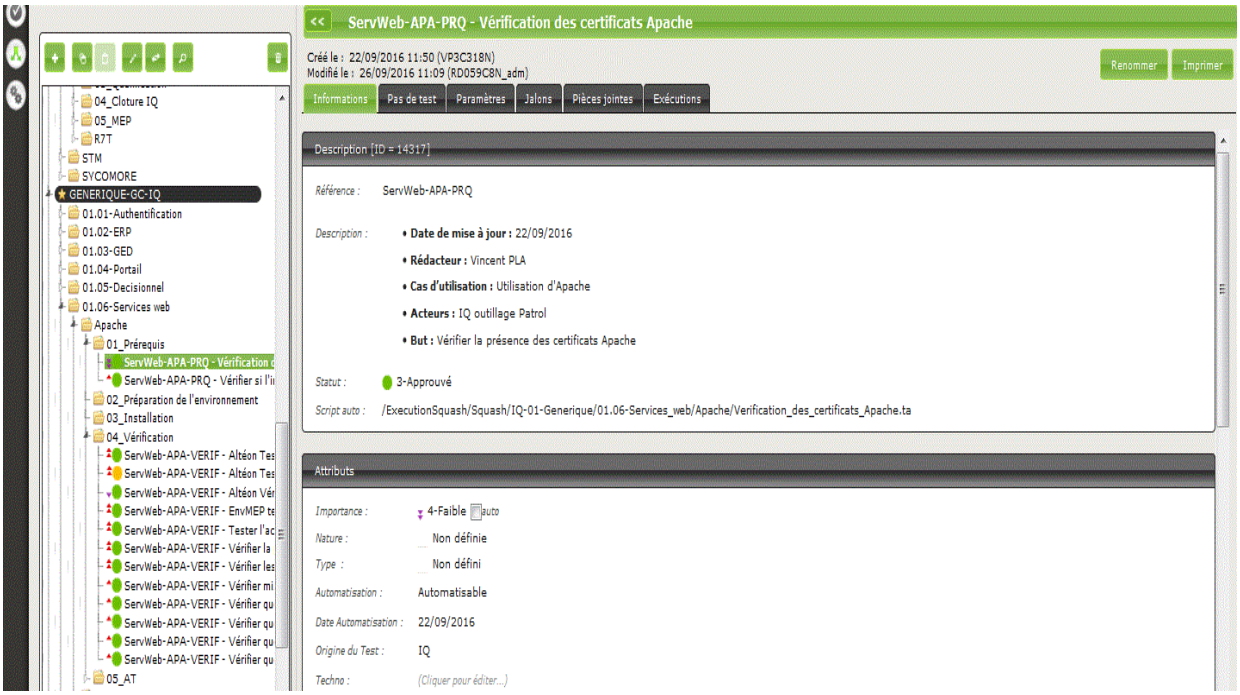


Figure 2.3 Illustration of the Squash platform

Figure 2.4 Illustration of the result of a test campaign

3 Realization of the mission

This chapter will detail the focal points of the traineeship: the context of the internship and the presentation of the team of which I was member, the presentation of the work plan, the problem and the pedagogical objectives, the progress of the stage, the tasks assigned to the trainee and their achievements, the statement of the problems encountered and their resolutions while performing the tasks and it ends with an assessment of the internship.

3.1 Context of the internship

From August 1st, 2016 to January 27th, 2017 I participated in the Enedis-IQ mission as an application integrator. This mission which took place inside the site of Atos Grenoble is attached to the Atos Integration department. Members of a team are about twenty people. My work progress was followed by Mr. David Marion as supervisor, Messrs. Damien Castinet, Julien Leclere and Lionel Audenis as service mangers, and Anne Geron as mentor.

3.2 Presentation of the Enedis-IQ team

The Enedis-IQ team is responsible for the integration and qualification of Enedis applications before they enter production. Applications are developed by "project teams" who often do not know the constraints of production. Our role is to make applications usable by processing technical installation, setting up monitoring, scheduling, backups and generic operating scripts to stop and restart the application easily. The IQ team also ensures compliance with Enedis application standards and application availability. Updating application documentation in compliance with all Enedis standards is within the scope of the mission.

3.2.1 Actors of the Enedis-IQ

The Application Integrator can interact throughout the application perimeter via IQs, work requests or incidents. The application integrator is responsible for performing the actions defined by the technical installation procedure that is used to update or upgrade the applications concerned. He works with the tooling integrator to set up the scheduling of the applications. During the course of an Integration-Qualification, it interacts with:

- ✓ The project for the technical points of the installation,
- ✓ Tool Integrators to set up Control-M scheduling applications,

- ✓ The industrialization team to process the scripting of the test case during the development of the acceptance test,
- ✓ The application manager to report on the progress of the project he is working on,
- ✓ Service managers for a daily report on the progress, difficulties or anomalies encountered.

The Tooling Integrator: It must develop the Control-M chains. The tooling integrator is in support of the application integrator during the IQ and interacts if the tooling is modified. They also handle work requests and incidents.

The industrialization team is in charge of automating the processus by means of scripts. It plays a major role during the making of the acceptance tests.

Service Manager and support of the Service Manager have the role of managing the team and representing it in front of the client. They assign the integrators to the various tasks, follow the IQ and communicate with the client in case of alert to go up. The *figure 3.1* below shows the Enedis-IQ organization chart

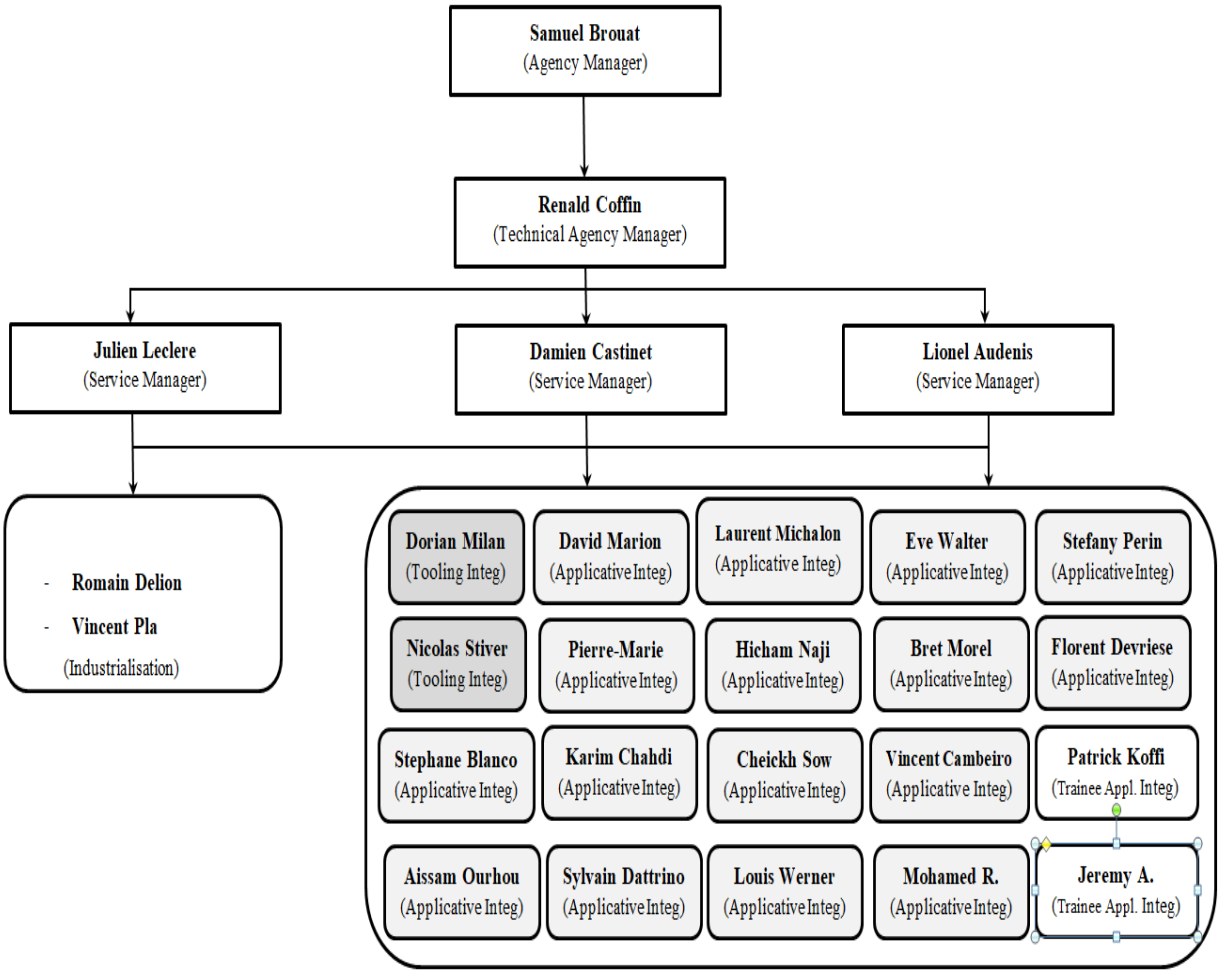


Figure 3.1 The Enedis-IQ organization chart [13]

3.2.2 Role and responsibilities

Table 3.1- Role and responsibilities of the Enedis-IQ actors

	Role	Responsibility
Integrators	• Technical expertise • Improve / optimize processes • Industrialization of test cases • Communication with the application manager	• Quality of deliverables • Compliance with OI-Enedis standards • Compliance with deadlines • Alert the service manager in case of incidents or drifts of the IQ
Industrialization team	• validation of test cases • Scripting test cases • Perform publication in the GITLAB repository	• Alerting the integrator in case of validation or non-validation of a test case • Respect the deadlines of scripting development
Service Managers	• Maintaining up-to-date planning • Follow-up of the activity • Communication with the client • Managing crisis situations • Validation of the IQ Plan	• Assignment of tasks to integrators • Ensure IQ deadlines are met • Customer Relationship • Organization of the team

3.3 Work and monitoring plans

3.3.1 Work plan

It has been defined as follows:

- Ensure the deployment of the evolutions of the platforms as well as the associated operating tool,

- Automate the deployment and design the industrialization of the acceptance tests activity of the various production tools and operating systems,

- Carrying out technological intelligence on UNIX.

To do this, I have evolved in a rich technical environment (UNIX, VMware ...) and complex (IBM PSeries, SAN, NAS, Blade Servers ...) requiring the use of an operating tool (Control-M Scheduler, Patrol Supervision, Networker backup).

Table 3.2 illustration of the internship road map

Actions	Description	Duration
Welcome the intern	Company presentation Team presentation	1 day
Presentation of the context	Presentation of the client Presentation of the role of the IQ team	2 days
Consideration of internal documents: • Reading charters of use of the computing resources • Consideration of the Integrator's Guide	• Signature of charters • Reading	2 days
Obtaining personal access to the Enedis domain	SESAME, Outlook, CACCIA, WERD, WERD-QUALIF, IS Actor, IQ Module, EMEDIA, PRODIS, CAPDOC, PEPITO, WIKI IQ, DMZ, Secure ID, Telephone line	1 week
Presentation of the tools: • Patrol • Newtest • Control-M • Pepito • Enedis tools • Adaje • EnvMep	• Overview of Application Monitoring • Presentation of scenarios and how they work • Presentation of the operating principle of the chains • Introduction to the incident management portal • Presentation of the various collaborative portals • Overview of automatic deployment • Presentation of the principle of functioning of the scripts	2 weeks
Presentation of manipulations • Follow-up of an IQE (IQ Express) • Follow-up of an Integration • Follow-up of a Qualification	Phase of appropriation	1 month
Technical operations achievements • Realization of an IQE with support • Realization of an integration with support • Realization of a qualification with support	Phase of realization	4 months

3.3.2 Monitoring plan

- The first few weeks were dedicated to presenting the mission context, tools and processes in place. Expected duration: one month
- Following by a phase of appropriation of the integration activities (working with a confirmed integrator). Expected duration: one month
- Then a phase of realization of the integration activities (supervised initially, then alone). Expected duration: four months

3.4 Problematic and objectives

3.4.1 Problematic

After obtaining the contract renewal of the applications qualification of the client Enedis and to render its service optimal, the IQ team has launched several projects. These main projects are:

- ✓ Industrialize and automate tasks,
- ✓ Bring added value through the acceptance tests.
- From the organizational point of view, the objective is to respond as quickly as possible to customer requests and requirements to ensure a quality of service.
- From the technical point of view, during the integration and qualification phases, it is necessary to make optimal use of the applications, by providing documentation and binaries which will allow time efficiency during production, and to minimize human error.

3.4.2 Objectives

The objectives are of two kinds:

- Pedagogical
  - Enable students to understand the work of a system integrator, its requirements and constraints,
  - Train students in the various architectures and technologies used,
  - Highlight its capacities of project coordination based on the acquired methods during its academic cursus.
- Personal
  - Work on a project from the beginning to the end. And also have autonomy, responsibilities.
  - Be competent in Application Integrator and System Administrator professions

3.5 Assigned tasks

As a junior application integrator, I had the following tasks:

- Realization of integration/qualification of various applications: realization, communication (reporting, anomalies, alerts)
- Optimizing the technical installation procedure: adding some helpful screenshots, add missing steps that have been forgotten by the project
- Updating the DEX
- Delivery of documentation
- Be a support to the outsourcing operator during production
- Completion of work requests and incidents
- Follow-up of the IQ tooling phase (scheduling, monitoring)
- Attend meetings with the client and third part
- Realization of the applications acceptance tests (drafting of test cases, and then push them to the industrialization team, execution)
- Implement the EnvMepV2 application management tool into applications that still working with the EnvMepV1

3.6 Detailed internship progress

August 2016: Phase of presentation (1 month)

From week_1 to week_2

- ✓ Welcome the intern, Presentation of the context, Consideration of internal documents
- ✓ Obtaining personal access to the Enedis domain

From week_3 to week_4

- ✓ Presentation of the tools (see *table 3.2* for details): important stage that shows the useful of each tool implemented
- ✓ Reading of the integrator's guide: this document is a necessary tool for each newcomer to the IQ team; it gives an overview of the client, and what the integrator work within the IQ team consists of.

September 2016: Phase of appropriation

This phase consists of following a confirmed integrator in the execution of the tasks assigned to him in order to become familiar with the activity

Week_1

- ✓ Application Integration of: GEC_v8.1.0

- ✓ Application Qualification of: GEC_v8.1.0

Week_2

- ✓ Applications checklist_ IQ of: SIRANO_v6.1.1 and SOLENN_v2.2
- ✓ Application Qualification of: DISCOMPTE_v16.2.2
- ✓ Application Integration and Qualification of: ATLAS_v3.1.1.e

Week_3

- ✓ Application Integration of: SIRANO_v6.1.1
- ✓ Acceptance tests of application: SOLENN_v2.2

Week _4

- ✓ Application Qualification of: SIRANO_v6.1.1
- ✓ Application Integration of: VICI_v2.0
- ✓ Implementation of the EnvMepV2 into: EPICEA

October 2016-Janvier 2017: Phase of realization

This part contains all the projects I worked on, so I conducted every project from beginning until the end.

Week_1

- ✓ Work request of: MERCURI
- ✓ Applications Qualification of: RUEDUSI_v16.10, SINJE_v6.1.3, VICI_v2.0
- ✓ Application checklist_ IQ of: SIAE_v1.2

Week_2

- ✓ Applications checklist_ IQ of: GARI_v6.1.1, IDS_v2.1.1
- ✓ Application Integration and Qualification of: GARI_v6.1.1
- ✓ Work request of: SIRANO
- ✓ Security training

Week_3

- ✓ Application checklist_ IQ of: ODIGO_v4.0.9.2
- ✓ Meeting attendance for: IDS
- ✓ Application Integration of: IDS_v2.1.1

Week_4

- ✓ Application Qualification of: IDS_v2.1.1
- ✓ Application Qualification of: GEC_v8.1.4.2
- ✓ Application checklist_ IQ of: FEC_v2.6

Week_5

- ✓ Application checklist_ IQ of: RPC_v2.3
- ✓ Application Integration and Qualification of: FEC_v2.6

From Week_6 until Week_7

- ✓ Application checklist_ IQ of: WIKIPROD_v2.1
- ✓ Application Integration of: RPC_v2.3

Week_8

- ✓ Application Qualification of: RPC_v2.3
- ✓ Training about the scheduler Control-M

Week_9

- ✓ Application checklist_ IQ of: LINCS_v4.2
- ✓ Work request: VICI, FEC

Week_10

- ✓ Application Integration and Qualification of: LINKYGED_v2.5, LINKYGED CONSTRUCTEUR_v1.7

From Week_11 to Week_13

- ✓ Development of the acceptance of: MySI, SIAE

From Week_14 to Week_16

- ✓ Application Integration and Qualification of: LINCS_v4.2

3.7 Execution of the tasks

This paragraph describes in a practical way the tasks I have performed. First, I describe what constitutes a work request, and then I do the same for the stages of the application integration, the application qualification, and the acceptance tests.

3.7.1 The work request

This is a one-time task requested by an actor from the OI-Enedis (the application manager, the project ...) and whose resolution requires the intervention of the integrator. This may include:

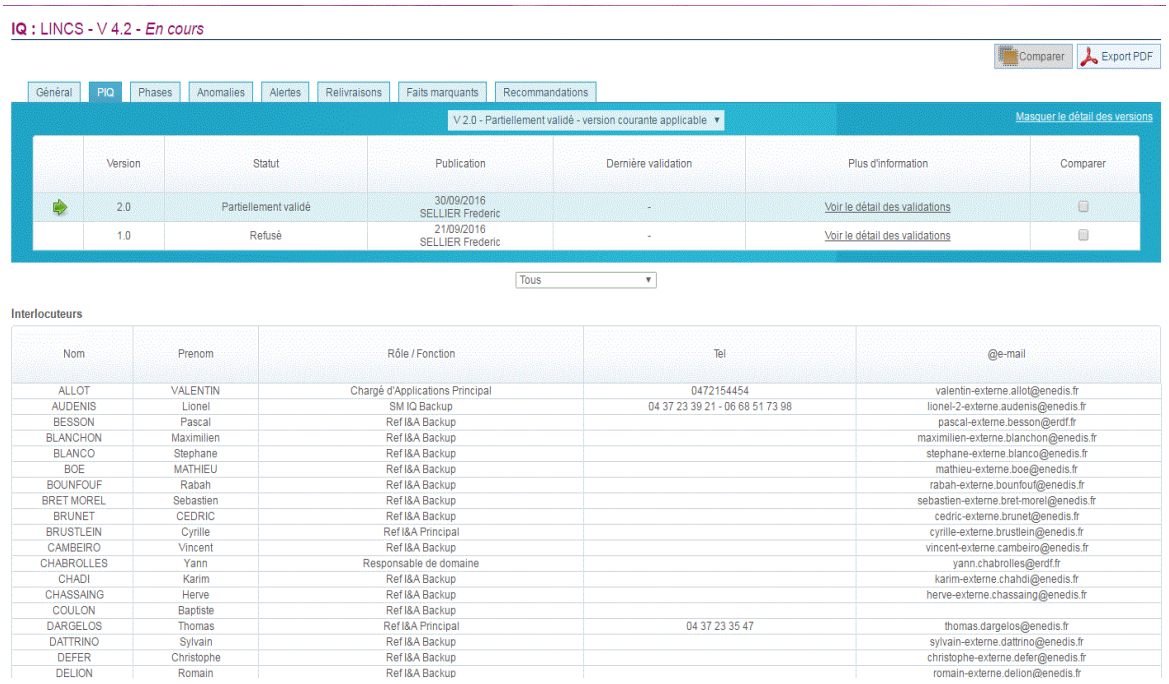
- Depositing a software stub on a given server
- Updating an authentication certificate on a web portal
- Making a backup of the database of the IQ environment and which will be used for the project
- Validating or not the deletion of a server clone made upstream of an IQ
- Stopping or starting up an application

3.7.2 Integration and Qualification of applications

This subsection refers to the administration of a farm of servers (the second part of the internship topic). I will describe the steps required for an application integration and qualification. Example: LINCS version 4.2 which is the upgrade version of the 4.1

3.7.2.1 Application integration

Step 1: Consulting the IQ plan. This represents the commitment for the IQ. It includes information such as the stakeholders of this IQ, the starting and ending dates of the IQ, the scope, etc. *The figure 3.2* shows the name, contact and email of the application manager, the project members, and the integrator for instance.



IQ : LINCS - V 4.2 - *En cours*

Comparer | Export PDF

Général | PIQ | Phases | Anomalies | Alertes | Relivraisons | Faits marquants | Recommandations

V 2.0 - Partiellement validé - version courante applicable

Masquer le détail des versions

Version	Statut	Publication	Dernière validation	Plus d'information	Comparer
2.0	Partiellement validé	30/09/2016 SELLIER Frederic	-	Voir le détail des validations	
1.0	Refusé	21/09/2016 SELLIER Frederic	-	Voir le détail des validations	

Tous

Interlocuteurs

Nom	Prenom	Rôle / Fonction	Tel	@e-mail
ALLOT	VALENTIN	Chargé d'Applications Principal	0472154454	valentin-externe.allot@enedis.fr
AUDENIS	Lionel	SM IQ Backup	04 37 23 39 21 - 06 68 51 73 98	lionel-2-externe.audenis@enedis.fr
BESSON	Pascal	Ref I&A Backup		pascal-externe.besson@erdf.fr
BLANCHON	Maximilien	Ref I&A Backup		maximilien-externe.blanchon@enedis.fr
BLANCO	Stephane	Ref I&A Backup		stephane-externe.blanco@enedis.fr
BOE	MATHIEU	Ref I&A Backup		mathieu-externe.boe@enedis.fr
BOUNFOUF	Rabah	Ref I&A Backup		rabah-externe.bounfouf@enedis.fr
BRET MOREL	Sebastien	Ref I&A Backup		sebastien-externe.bret-morel@enedis.fr
BRUNET	CEDRIC	Ref I&A Backup		cedric-externe.brunet@enedis.fr
BRUSTLEIN	Cyrille	Ref I&A Principal		cyrille-externe.brustlein@enedis.fr
CAMBEIRO	Vincent	Ref I&A Backup		vincent-externe.cambeiro@enedis.fr
CHABROLLES	Yann	Responsable de domaine		yann.chabrolles@erdf.fr
CHADI	Karim	Ref I&A Backup		karim-externe.chahdi@enedis.fr
CHASSAING	Herve	Ref I&A Backup		herve-externe.chassaing@enedis.fr
COULON	Baptiste	Ref I&A Backup		
DARGELOS	Thomas	Ref I&A Principal	04 37 23 35 47	thomas.dargelos@enedis.fr
DATTRINO	Sylvain	Ref I&A Backup		sylvain-externe.dattrino@enedis.fr
DEFER	Christophe	Ref I&A Backup		christophe-externe.defer@enedis.fr
DELION	Romain	Ref I&A Backup		romain-externe.delion@enedis.fr

Figure 3.2 IQ plan of LINCS version 4.2

Step 2: Run the daily control-M chain. This ensures that the application is running. *Figure 3.3* illustrates the control-M chain of LINCS that is run daily. In the white background of the left position, there is information such the execution date “170112 means 12-january-17”, the jobs that have been run successfully in green label and those that failed in red.

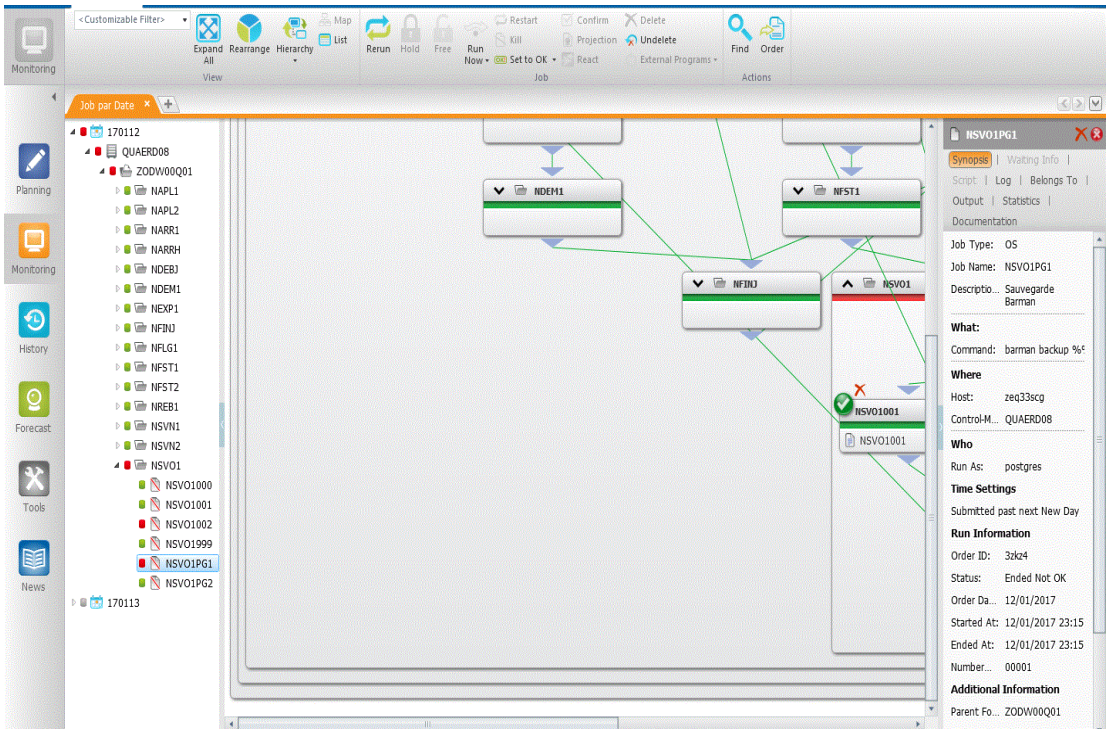


Figure 3.3 Illustration of the daily Control-M chain of LINCS

Note: When IQ starts up, make sure that the application Control-M chain is not running, to avoid supervisory warnings.

Step 3: Thanks to the monitoring console, I have sorted the application alerts out. The INFRA alerts were reported to the project team. In the *figure 3.4*, we observe that there is no more alert (if any, should be appeared in red) appearing.

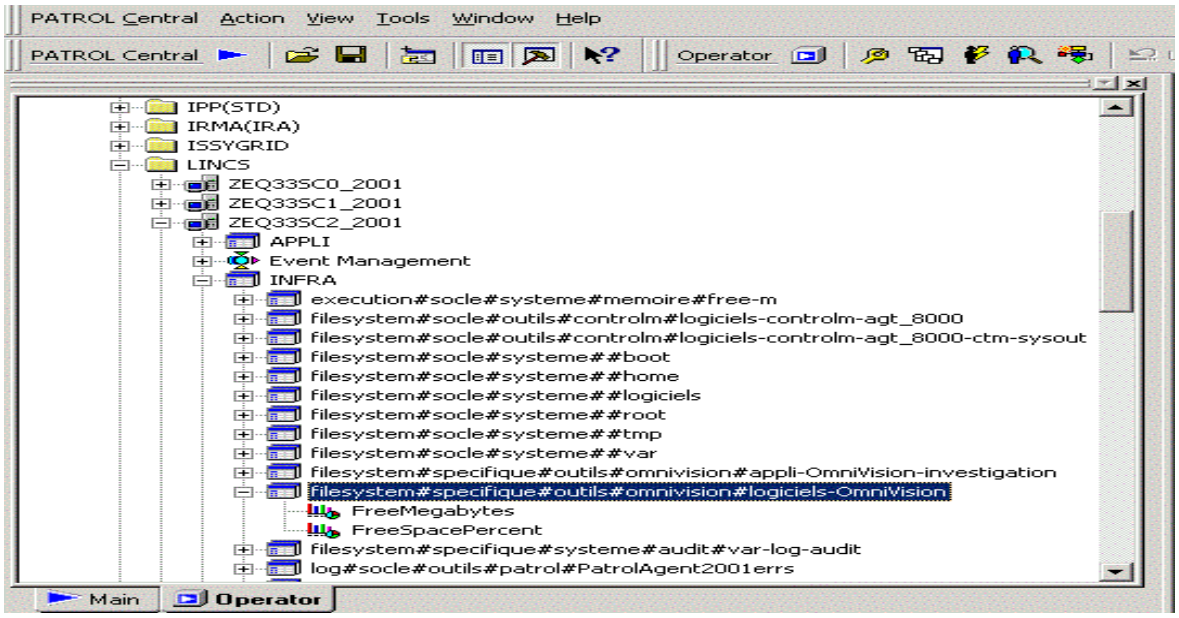
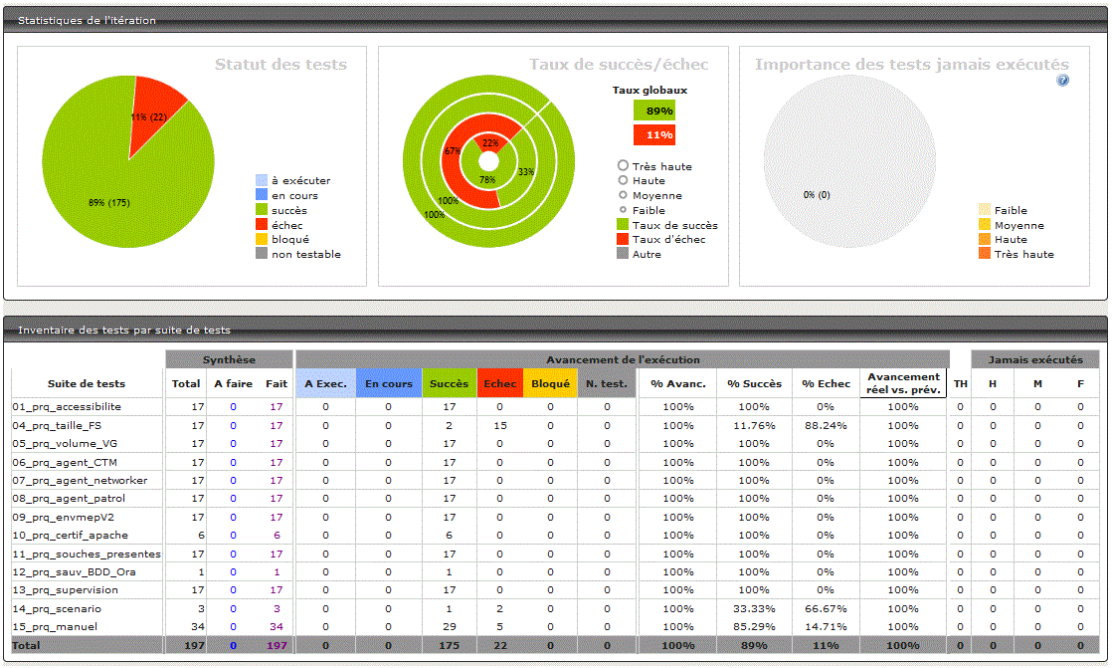


Figure 3.4 Monitoring alerts illustration of LINCS

Step 4: Evaluate the prerequisites of the environment before any integration action. I thus carried out a prerequisite test campaign illustrated by the *figure 3.5* below. This shows tests on availability, file system size, supervision, etc. In the final result of this campaign, it is observed that 175 tests have been run successfully while 22 failed. The failed tests have been reported to the teams in charge.

Note: No test should remain in status "To execute", "In progress" or "blocked"



Suite de tests	Synthèse			Avancement de l'exécution										Jamais exécutés			
	Total	A faire	Fait	A Exec.	En cours	Succès	Echec	Bloqué	N. test.	% Avanc.	% Succès	% Echec	Avancement réel vs. prév.	TH	H	M	F
01_prq_accessibilite	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
04_prq_taille_FS	17	0	17	0	0	2	15	0	0	100%	11.76%	88.24%	100%	0	0	0	0
05_prq_volume_VG	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
06_prq_agent_CTM	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
07_prq_agent_networker	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
08_prq_agent_patrol	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
09_prq_envmepV2	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
10_prq_certif_apache	6	0	6	0	0	6	0	0	0	100%	100%	0%	100%	0	0	0	0
11_prq_souches_presentes	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
12_prq_sauv_BDD_Ora	1	0	1	0	0	1	0	0	0	100%	100%	0%	100%	0	0	0	0
13_prq_supervision	17	0	17	0	0	17	0	0	0	100%	100%	0%	100%	0	0	0	0
14_prq_scenario	3	0	3	0	0	1	2	0	0	100%	33.33%	66.67%	100%	0	0	0	0
15_prq_manuel	34	0	34	0	0	29	5	0	0	100%	85.29%	14.71%	100%	0	0	0	0
Total	**197**	**0**	**197**	**0**	**0**	**175**	**22**	**0**	**0**	**100%**	**89%**	**11%**	**100%**	**0**	**0**	**0**	**0**

Figure 3.5 Prerequisites campaign of LINCS

Step 5: Check the deliverables as input and compare this delivery with what is advertised in the IQ plan

Step 6: Fill out the checklist illustrated in *figure 3.6*

Note: According to the IQ, it may be required to clone servers. In this case, a work request for the cloning of the targeted servers should be done.

The *figure 3.6* shows the requirements that need to be checked before starting an integration/qualification.

Informations IQ				Labels		
				Livrables entrée d'IQ	IQ	Livrables sortie d'IQ
Application	LINCS	Date de Début	09.01.2017			
Version	4.2	Date de Fin	20.01.2017			
Module	IQ Standard	Date de Préprod	23.01.2017			
Pesée	Complexe	Date de Prod	26.01.2017			

Critères obligatoires

Action	Acteur	Etat	Conformité	Mesure	Date cible	Date de réalisation	Commentaires (obligatoire si Etat à Non)	Label livrables	
Pré-requis documentaires								En	Sc
Vérifier que le consuel et/ou DAT soit validé et conforme	CA						Alerter si DAT/CONSUEL non validé ou non conforme		
Le PIQ existe-t-il et est-il conforme?	CA						Délai de prévenance à mesurer (via extraction IQ)		
La pesée IQ est-elle conforme?	CA / Projet / IQ	Réalisé			30.09.16	04.01.17	Légère / Standard / Complexe		
RT & SM IQ ont-ils validés le PIQ?	CA						Envoyer une relance si nécessaire		
Communiquer la PTI à l'IQ (version draft ou finale)	CA						Hors module 4 - Alerter si PTI non reçue		
Le DAT a-t-il été transmis à l'IQ?	CA / Projet							X	
Le DEX a-t-il été transmis à l'IQ et à IPI?	IPI							X	
Vérification de la conformité de l'environnement avec le DEX	IQ	Réalisé				04.01.17	Non conforme: absence des parties Postgres et Mobilité	X	
Vérification de la conformité de l'environnement avec le DEX	IPI							X	
Vérification de la conformité de l'environnement avec le DAT / Consuel	IQ	Réalisé				04.01.17		X	
Vérification de la conformité de l'environnement avec le DAT / Consuel	IPI							X	
Pré-requis techniques									
Vérifier la livraison d'une nouvelle machine	CA						J-1 avant démarrage IQ - Alerter si les pré-requis sont KO	X	
Vérifier la volumétrie VG	CA						J-1 avant démarrage IQ	X	
Vérifier la taille des FS	CA						J-1 avant démarrage IQ	X	
Vérifier les souches installées	CA						J-1 avant démarrage IQ - Alerter si les pré-requis sont KO	X	
Vérifier l'ouverture des flux	CA						J-1 avant démarrage IQ- Alerter si les pré-requis sont KO	X	
Demander une sauvegarde de la base de production	CA								
Vérifier que les certificats soient livrés	CA							X	
Vérifier la création des comptes techniques	CA							X	
Vérifier les connexions caccia aux machines	IQ	Réalisé				03.01.17	Alerter si les pré-requis sont KO		
Disposer d'un compte Newtest valide en IQ	IPI								
Vérifier le(s) scénario(s) Newtest	IPI								

Figure 3.6 Illustration of an IQ checklist

Step 7: Phase opening of the first forward step in the IQ module (*cf. figure 3.7*)

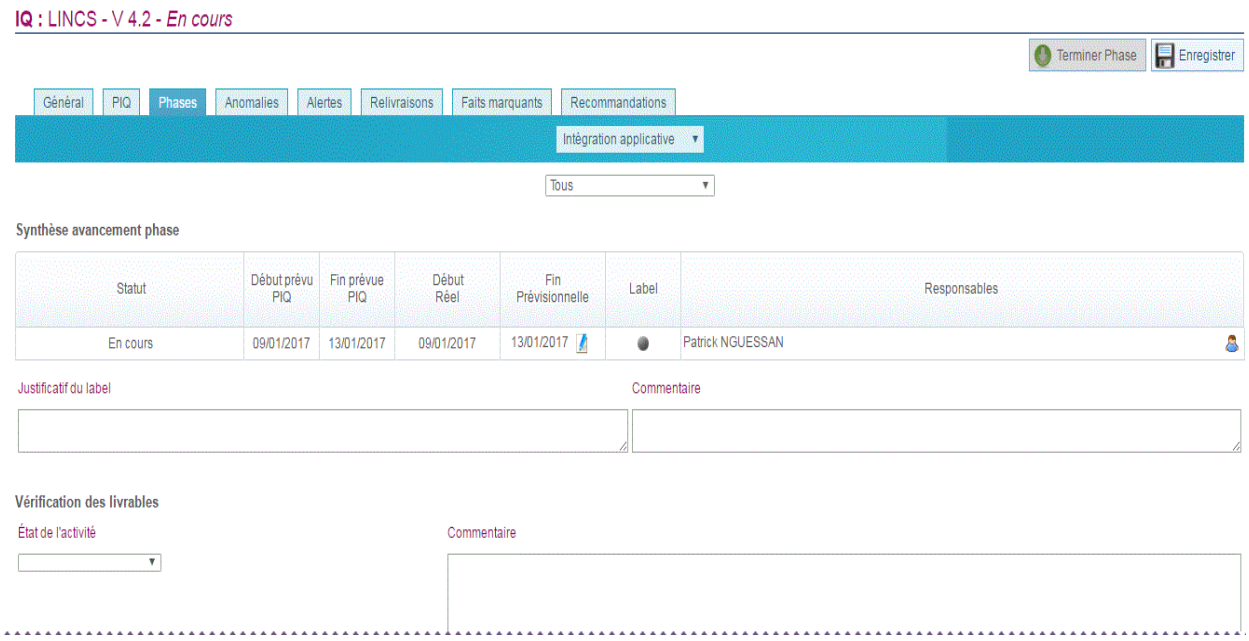


Figure 3.7 Illustration of the phase opening of the application integration

Figure 3.7 shows that I have opened the phase on 09/01/2017 as expected by the IQ Plan and the status is “In Progress”.

Step 8: Perform all actions described in the technical installation procedure (*cf. figure 3.8*) and correct any anomalies that may occur.

Afin de préparer un retour arrière éventuel, veuillez effectuer les actions suivantes sur les machines indiquées.

- ❑ Sauvegarde Liferay Staging

Serveurs :	Temps d'exécution :
<LIFERAY_STAGING>	2 min

Taper les commandes suivantes:

```
<NNI># sudo su - webadm
```

Supprimer le backup de l'ancienne MEP s'il existe, c'est un tar dans le répertoire /appli/projects/LINCS.tar :

```
<webadm># cd /appli/projects/
<webadm># rm -f LINCS.tar
```

Créer une nouvelle sauvegarde :

```
<webadm># tar cvf LINCS.tar LINCS --exclude "LINCS/migration"
```

- ❑ Sauvegarde Liferay Live 1 et Liferay Live 2

Serveurs :	Temps d'exécution :
<LIFERAY_LIVE_1> <LIFERAY_LIVE_2>	5 min

Taper les commandes suivantes:

```
<NNI># sudo su - webadm
```

Supprimer le backup de l'ancienne MEP s'il existe, c'est un tar dans le répertoire /appli/projects/LINCS.tar :

```
<webadm># cd /appli/projects/
<webadm># rm -f LINCS.tar
```

Créer une nouvelle sauvegarde :

```
<webadm># tar cvf LINCS.tar LINCS
```

Figure 3.8 Technical installation procedure of LINCS version 4.2

Step 9: At the end of the application integration also called first forward step, the integrator must notify:

- The project team to obtain the technical validation which consists of seeing if the actions performed by the integrator technically reflects the expected result.
- The business team to obtain the functional validation to see if the upgraded version corresponds to the expectations of the business.

Step 10: I have realized the reversing step (*cf. figure 3.9*). It is a question of returning to the version before the first step forward. It is a practice guaranteeing the restoration of the application in the case of problems encountered during the production-start.

Retour arrière LINCS

Nous allons donc restaurer toutes les sauvegardes faites au paragraphe 3.1.

4.4 Mise en maintenance Apache Staging

Serveurs :	Temps d'exécution :
<APACHE_STAGING>	A remplir en phase de QLF

<NNI># sudo su - webadm

<webadm># touch /appli/projects/LINCS/apache_2.2.24/error/maintenance.enable

4.5 Arrêt Liferay Staging

Serveurs :	Temps d'exécution :
<LIFERAY_STAGING>	A remplir en phase de QLF

<NNI># sudo su - webadm

<webadm># stopLiferay

4.6 Restauration de la sauvegarde Liferay Staging

Serveurs :	Temps d'exécution :
<LIFERAY_STAGING>	A remplir en phase de QLF

Figure 3.9 Illustration of the reversing steps

Note: After this step, a functional validation is expected to validate the proper execution of the reverse.

3.7.2.2 Application qualification

The application qualification is also named second forward step. It represents the confirmation of the upgrade version as desired. Below are the different steps.

Step 1: Phase opening of the qualification step in the IQ module (*cf. figure 3.10*). The exhibit shows that I am about to open the phase. The opening will be done by clicking the button “commencer phase”.

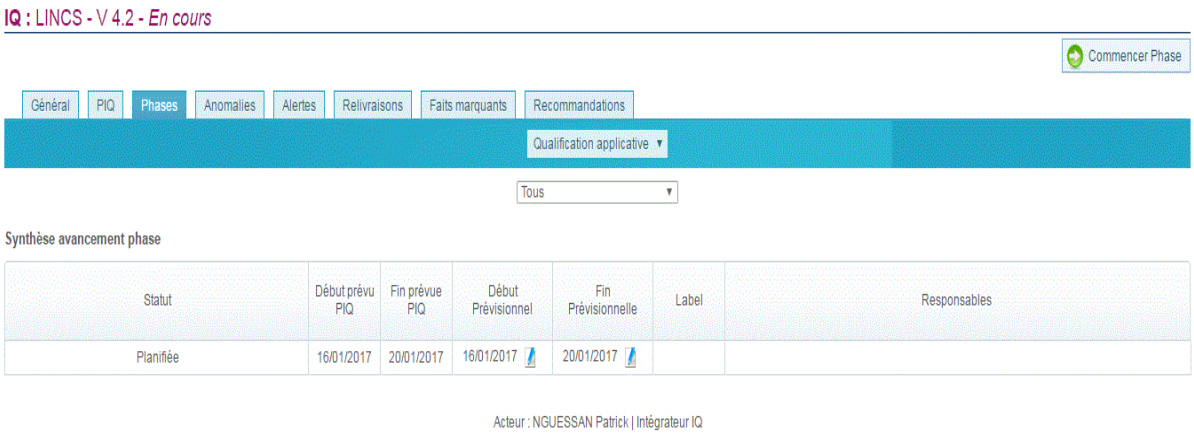


Figure 3.10 Illustration of the phase opening of the application qualification

Step 2: It is a question of performing the same actions as the first forward step, in a more linear manner and recording the actions timing in order to calibrate the production-start duration.

Note: After this step, a functional validation is expected to validate the proper execution of the second forward step.

Step 3: Delivery of the deliverables to the production outsourcing operator, so that he can start production (*cf. figure 3.11*)

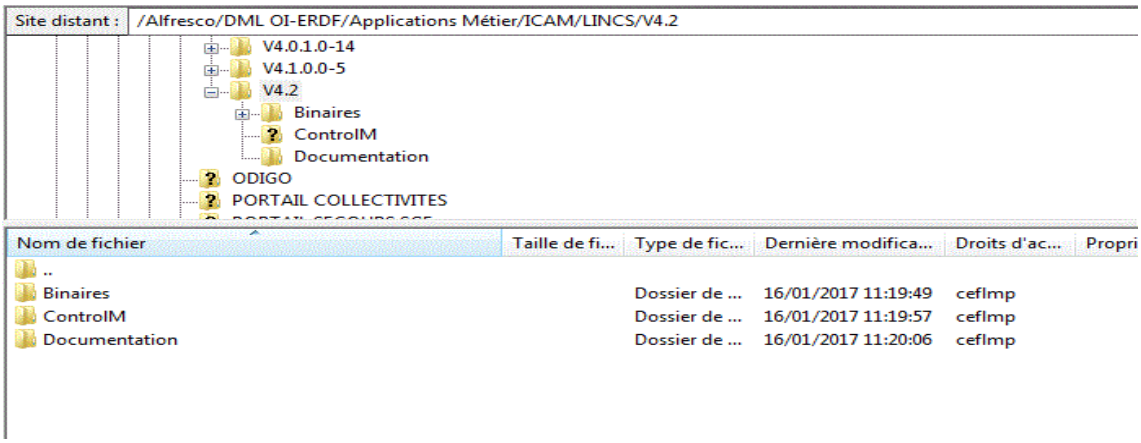


Figure 3.11 Illustration of the delivery

3.7.3 Acceptance test

The acceptance test is to carry out an automated tests campaign based on the application components, the exploitation, and the tooling. The objective is to maintain

the application in a good functional state through specific checks. The necessary steps for the acceptance test of the SIAE application will be described in detail.

Step 1: Application support. It starts with the inventory of the products that make up the application. Our example in the red box includes Apache, JBoss, Mersi, Oracle, and PROFTP. Operational testing is for EnvMEPv2, Scheduling, Backup, and Monitoring (*cf. figure 3.12*).

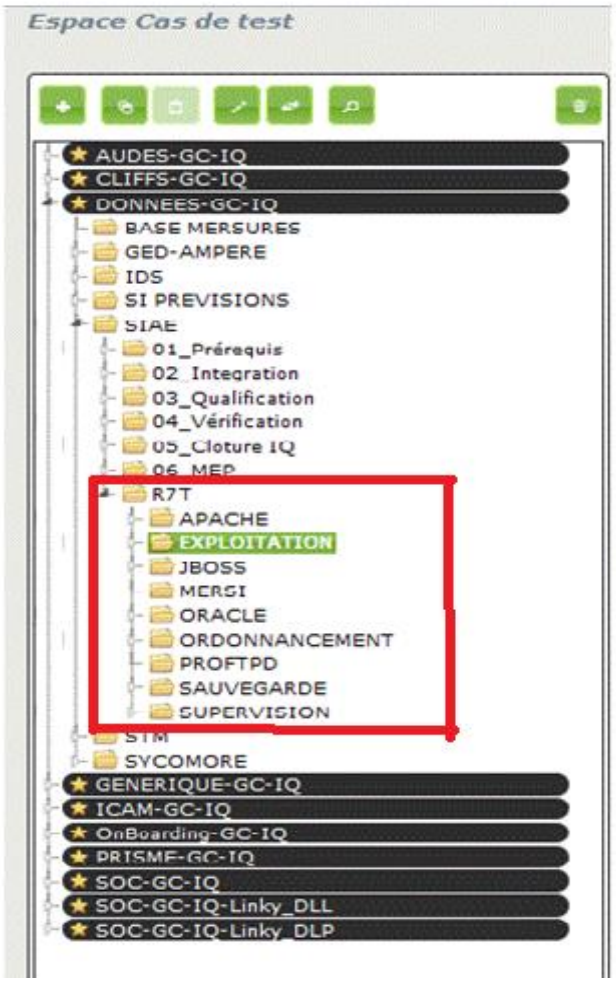


Figure 3.12 Content of the SIAE acceptance test

Step 2: Creation of test cases. The role of the integrator at this level is to identify plausible test cases and edit them in the Squash test platform. A test case can consist of several steps (*cf. figure 3.13*).

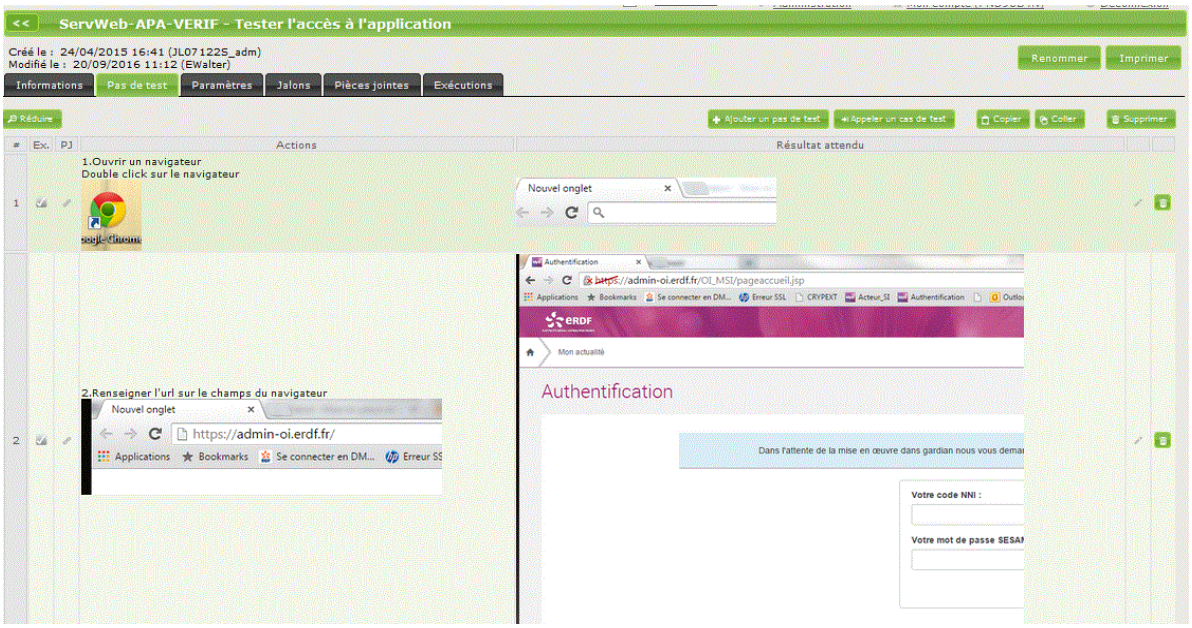


Figure 3.13 Illustration of a test case

The example above depicts the test case called "The application access test". It consists of two test steps: 1) open a browser and 2) fill in the URL (Uniform Resource Locator) in the browser field.

Step 3: Validate the different test cases. It is a matter of putting the test cases at the disposal of the industrialization team to validate the test case.

Step 4: Scripting and Publishing. The SQUASH referents carry out the scripting of the relevant test cases in order to allow their exploitation in the upcoming campaign. Once the script is functional, the publication is done in GITLAB as illustrated in *figure 3.14* below.

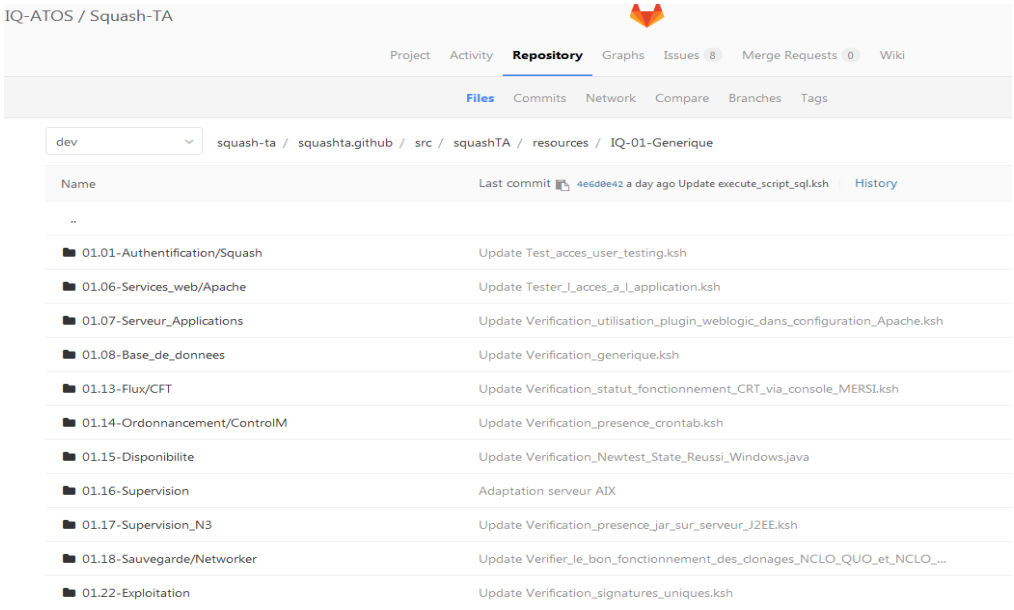


Figure 3.14 Script repository in GITLAB

Step 5: Defining the test case parameters. The parameters are taken into account by the script related to the test case. The example illustrated by *figure 3.15*, the current parameter is the “Target”: the host server (s). Concretely the test case highlighted by green color (on the side of the figure below) will have as parameters, servers zeq335b0, zeq335b1, zeq335b2, zeq335b3.

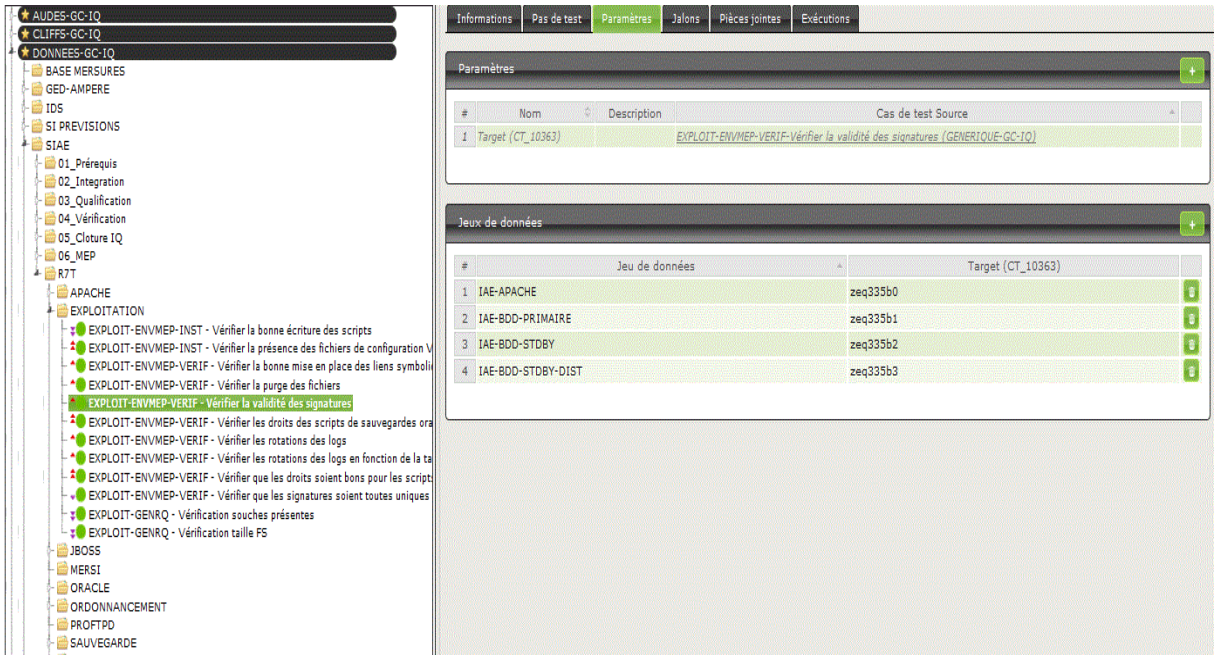


Figure 3.15 Illustration of the test case parameters definition

Step 6: Creation of the acceptance test campaign. It consists of regrouping all the test cases that will make up this campaign. *Figure 3.16* illustrates the content of a test campaign.

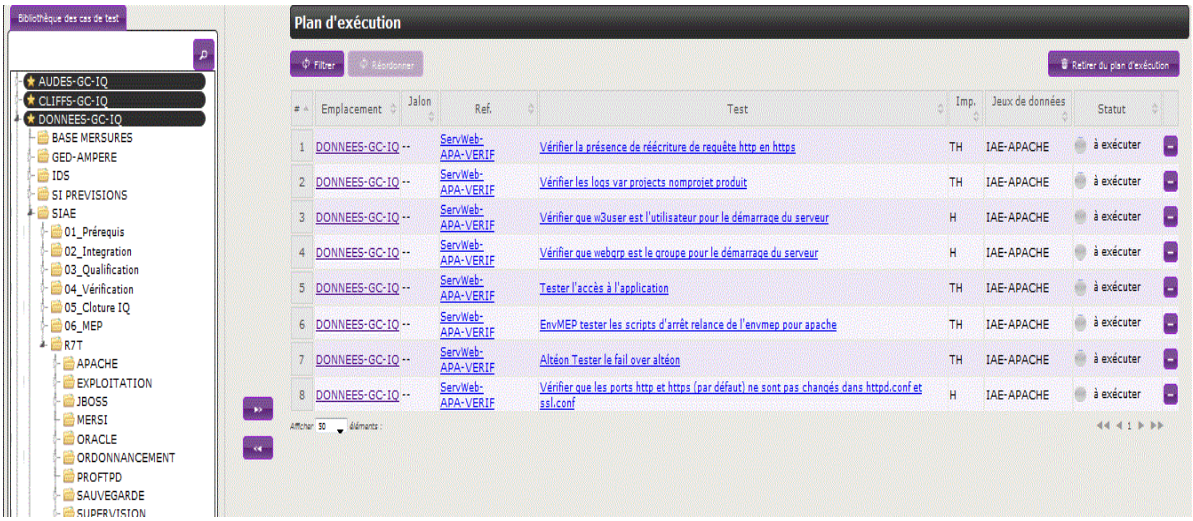


Figure 3.16 Content of the execution plan for an acceptance test campaign

Step 7: Run the campaign tests using the red box (*cf. figure 3.17*) and visualize the evaluation of the acceptance test (*cf. figure 3.18*)

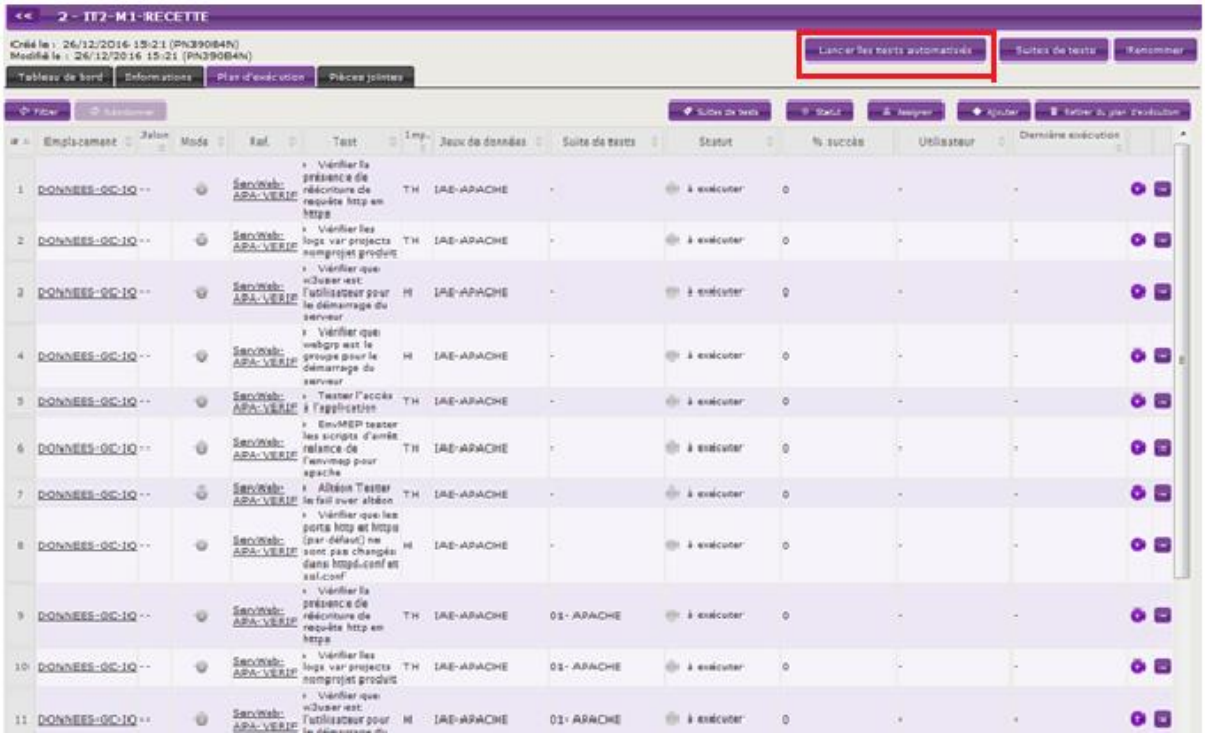
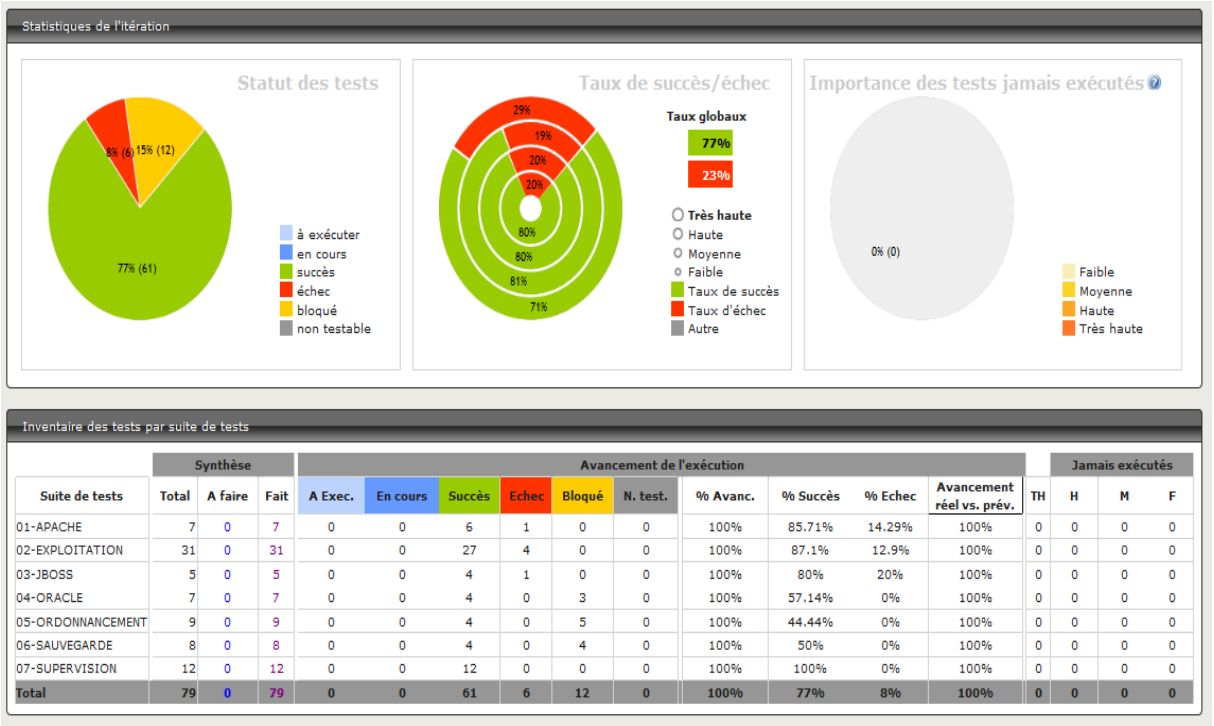


Figure 3.17 Running automate campaign tests



	Synthèse			Avancement de l'exécution										Jamais exécutés			
Suite de tests	Total	A faire	Fait	A Exec.	En cours	Succès	Echec	Bloqué	N. test.	% Avanc.	% Succès	% Echec	Avancement réel vs. prév.	TH	H	M	F
01-APACHE	7	0	7	0	0	6	1	0	0	100%	85.71%	14.29%	100%	0	0	0	0
02-EXPLOITATION	31	0	31	0	0	27	4	0	0	100%	87.1%	12.9%	100%	0	0	0	0
03-JBOSS	5	0	5	0	0	4	1	0	0	100%	80%	20%	100%	0	0	0	0
04-ORACLE	7	0	7	0	0	4	0	3	0	100%	57.14%	0%	100%	0	0	0	0
05-ORDONNANCEMENT	9	0	9	0	0	4	0	5	0	100%	44.44%	0%	100%	0	0	0	0
06-SAUVEGARDE	8	0	8	0	0	4	0	4	0	100%	50%	0%	100%	0	0	0	0
07-SUPERVISION	12	0	12	0	0	12	0	0	0	100%	100%	0%	100%	0	0	0	0
Total	**79**	**0**	**79**	**0**	**0**	**61**	**6**	**12**	**0**	**100%**	**77%**	**8%**	**100%**	**0**	**0**	**0**	**0**

Figure 3.18 Illustration of the result of the SIAE application campaign

Note: No test cases should remain in «To execute" and "In Progress" status. As shown in figure 3.17, the result of this acceptance test campaign gives a success rate of 77% corresponding to a green label.

3.8 Problems solved during the IQ phases

During the realization of the mission, I encountered a few problems that I have been able to find or help to find solutions:

- Issue 1 (RPCv2.3): RPC_BO_v2_3.hsh script failed to execute to create a system file, to mount it and then use it in the rest of the script.

✓ Resolution 1: Insert a 10-second timer between the end of mounting the system file and its use

- Issue 2 (FECv2.6): FEC user failed to run a script to replace the activity reports of year N with that of year N + 1.

✓ Resolution 2: Assignment of execution rights to this user

- Issue 3 (LINCSv4.2): LINCS application portal Error after passing from HTTP to HTTPS.

✓ Resolution 3: The listening ports of the managed instances and the domain instance have been swapped, so we put at the right place.

- Issue 4: This concerns some difficulties in addressing high-level Patrol alerts

✓ Resolution 4: To remedy these alerts:

- Option 1: I do research in documentation materials like DEX, WIKI IQ, for an in-depth understanding of the application is being exploited
- Option 2: I also do research on the internet.
- Option 3: I contact the patrol referrer of the mission
- Option 4: I reach the Enedis technical support, via a work request to deal with the problem.

4 The assessment of the internship

4.1 Results

All applications on which I made integration and qualification have passed production. Feedbacks are all positive and applications are available to end users.

4.2 Professional skills acquired

I have acquired several skills in different fields:

- ✓ Operating systems management: Redhat, AIX, Windows
- ✓ Web server management: Apache
- ✓ Application servers management: JBoss, Tomcat, Weblogic
- ✓ Databases management: Oracle, PostgreSQL
- ✓ Tools: Patrol, Control-M, Networker

I also developed my autonomy and my ability to manage a project from beginning to end, by using the IQ documentation, and following the advice of my advisor, and IQ team members.

From now, I master the process of integration and qualification of the applications. I corrected some abnormalities on servers and I draft documentation.

4.3 Interpersonal skills

On the relational level, I was led to communicate over the course of the water on the progress of each IQ. This is done in order to keep track of all the actions carried out and to succeed in capitalizing them. This project was an opportunity to reveal my abilities. I have been seriously involved so that this project will succeed in all aspects.

4.4 Time efficiency

I was well supervised by my advisor. This allowed me to well manage the time. I learned to set myself a roadmap that allowed me to balise my IQs in order to complete them on time. Then I became more autonomous in the management of my IQs, and choose the priority tasks.

4.5 Working at Atos

Working in a large IT service provider company involves great self-confidence and abilities. The work must be correct and optimal. I immediately realized that I had to be more rigorous in carrying out the tasks assigned to me. Changes in positions are important.

5 Conclusion

The ISD has the difficult task of matching the IT system of its organization and the changes in the business (whether managerial, structural, commercial or technological) that occur internally.

In this regard, Enedis is working with internal and external service centers to ensure and meet this challenge. In this scope, I have integrated one of them, namely the Integration-Qualification (IQ) provided by the Atos entity. My integration in this mission has been of great benefit on several points:

- ✓ Educational: by acquiring new knowledge on technologies not yet explored in the academic curriculum (administration of AIX systems or Oracle databases, PostgreSQL, Control-M scheduling tool, Patrol supervision tool, etc.)
- ✓ Professional: by putting into practice certain knowledge previously acquired and also by practicing the profession of the integrator which enabling to develop certain capacities such as self-study, anticipation, and precision.
- ✓ Personal: through a relational development (sense of communication between the stakeholders of the qualification of an application), and human by self-control in situations of stress and proven complexity.

These are 19 applications in total that I have worked on and this represents 33 managed servers. This internship has made me progress on the qualities necessary for the profession of integration and application qualification namely:

- ✓ Self-education (learning to seek complementary information about a technology through various sources)
- ✓ Technical and technological monitoring
- ✓ The sense of anticipation: once assigned to an IQ, check before it begins the conformity of all the deliverables (installation procedure, binary, batch, script which can be browsed to understand what it allows to do, etc.)
- ✓ Self-control to manage stressful situations related to a bad evolution of an IQ
- ✓ Be methodical to capitalize on the experiences encountered.

I will conclude by saying that this experience in the workplace has been particularly enriching and insightful, both from a human and a technical point of view. I realized that this internship at Atos has been totally beneficial to me in terms of all these skills contributions.

Part 2: Improving TCP Performance in Underwater Wireless Sensor Networks

1 Introduction

The planet Earth is covered with 70.71% water for 29.29% of the land, making the seaway a vast natural field to explore. Scientists (oceanographers, seismologists, meteorologists, etc.), have proved that many natural phenomena find their source in the seabed or other watercourses. This has prompted a great deal of research in this natural environment, in addition to the submarine navigation which developed during the World War 2. First attempts were directed toward the underwater telephone in 1945. Very soon, it appeared a wide range of applications in this field, such as in [1]:

- Environmental monitoring including pollution monitoring (chemical, biological, and nuclear), ocean current and wind monitoring, and biological monitoring such as tracking of fish or microorganisms, weather forecast, climate change, predicting the effect of human activities on marine ecosystems.
- Undersea explorations including detection of oil fields or reservoirs, determination of routes for laying undersea cables, and assists in the exploration for valuable minerals.
- Seismic monitoring and disaster prevention provide warnings to coastal areas or study the effects of submarine earthquakes (seaquakes).
- Equipment monitoring enabling remote control and temporary monitoring of expensive equipment immediately after the deployment, to assess deployment failures in the initial operation or to detect problems.
- Assisted navigation to identify hazards on the seabed, locate dangerous rocks or shoals in shallow waters, mooring positions, submerged wrecks, and to perform bathymetry profiling.
- Distributed tactical surveillance includes collaboratively monitoring areas for surveillance, reconnaissance, targeting, and intrusion detection.
- Mine reconnaissance to perform the rapid environmental assessment and detect mine-like objects.
- Military and homeland security applications which include securing and monitoring port facilities or ships

These potential applications will be made viable by enabling communications among underwater devices including unmanned vehicles for environments that are restrictive for

humans. Underwater sensor networks make it possible. There has been an increased interest in building underwater wireless sensor networks (UWSN) due to the fact that, these networks have the ability to monitor, investigate, and track underwater occurrences that are important to our security and offshore exploration. However, many characteristics (that will be discussed in subparagraph 2.1.2) of these networks make their construction challenging, and difficult to implement. As a result, many studies have been conducted even though most of them were physical, data link and network oriented, leaving the transport layer less explored in this type of communication. Thus the willingness to lead a study in this field with a reliable transport protocol is motivated by this aspect. However, traditional transmission control protocol (TCP) cannot work effectively in an environment favorable to multiple and successive losses such as UWSN. In fact, TCP systematically reduces the size of the sending window (hence its throughput) whenever a packet loss occurs. Thus the losses in burst would lead to a closure of the connection which is an obstacle to communication. Nevertheless, TCP variants (discussed in 2.2.3) with more advanced mechanisms could be useful for our research.

1.1 Context

There has been an increasing interest in the development of Underwater Wireless Sensor Networks (UWSNs) in the last years. The first attempts to analyze UWSN behavior were based on the mature technology developed during the last decade in terrestrial wireless sensor networks (TWSNs) [2]. Several researches have allowed the concept to progress despite the harsh environment of the underwater communication channel, which is characterized by a high attenuation, a great deal of noise in the lower end, a limited bandwidth, Doppler effect, multipath and high propagation delay [3][4][5][6]. However, most of the improvements are related to physical, data link and network layers, making the transport-layer of UWSN totally unexplored area [3] .TCP improvements in UWSNs may represent a great challenge of study and eventually guarantee a reliable data transmission protocol.

1.2 Motivation

The TCP is, by far, one of the most important transport protocols that are used by many of the Internet's popular applications, including the WorldWideWeb (WWW), File Transfer Protocol (FTP), and some streaming media applications. Therefore, there is a strong motivation to consider TCP performance in UWSNs. It is well known that the TCP may suffer from severe performance degradation in wireless networks. This problem may be more severe in UWSNs. Thus, if the TCP is not carefully considered in UWSN, the

perceived TCP performance degradation may impede the success of UWSN technologies. Due to the TCP improvement in wireless networks, UWSNs may have a basis for the improvement of its performance.

Considering the fact that the subject is less explored combined with the difficulties in the acquisition of materials, we will carry out simulations to record the possible effects of an improved TCP in UWSN.

1.3 Statement of the problem

The traditional TCP is unsuited for the underwater environment since the flow control functionality is based on a window-based mechanism that relies on an accurate estimation of the round trip time (RTT), which is twice the end-to-end delay from source to destination. The underwater RTT can be modeled as a stochastic variable. Furthermore, the high variability of the RTT would make it hard to effectively set the timeout of the window-based mechanism that most current TCP implementations adopt. Moreover, Rate-based transport protocols seem also unsuited for this challenging environment. In fact, although they do not adopt a window-based mechanism, they still rely on feedback control messages sent back by the destination to dynamically adapt the transmission rate, i.e., to decrease the transmission rate when packet loss is experienced or to increase it otherwise. The high delay and delay variance can thus cause instability in the feedback control. Most TCP implementations, which are designed for wired networks, assume that congestion is the only cause for packet loss. Due to this assumption, when a packet loss occurs, they reduce the transmission rate to avoid injecting more packets in the network. Conversely, in UWSNs as well as in terrestrial wireless networks, it is important to discriminate losses due to impairments of the channel from those caused by congestion. When congestion is the cause of the packet loss, the transmission rate should be decreased to avoid overwhelming the network; while in the case of losses due to bad channel quality, the transmission rate should not be decreased to preserve throughput efficiency. Since in wireless sensor networks (WSNs) in general, packet losses are due to the poor quality of the wireless channel, sensor failure and network congestion, it seems important to devise a technique or approach to overcome these aspects.

1.4 Objective

The purpose of this dissertation is to show that relevant improvements of TCP performance are possible in an underwater environment. To do this, two approaches can be identified:

- ✓ One is the proposal of an algorithm that can efficiently manage the underwater environment,
- ✓ The other is based on the identification of changes or additions that can be made to achieve the goal.

To achieve this, we should be able to carry out a number of sub-objectives:

- Build suitable scenarios,
- Set the system up,
- Implement the reached solutions,
- Analyze the performance regarding the measurements criteria,
- Highlight possible advances if any.

The rest of this thesis is organized as follows. Chapter 2 provided an overview of UWSN and transport control protocol. We discuss underwater sensor structure, the channel characteristics, communication architectures and the layered model. We also present UWSN challenges. Then, we provide a review of TCP, its variants and discuss some related works. Chapter 3 introduces U-Newreno which stands for our solution to improve TCP in UWSN. It also presents the simulation and performance evaluation. We discuss various simulation tools for underwater sensor networks, simulations and analyze the results. The thesis is completed by the conclusion and future direction in chapter 4.

2 Background

Although the aquatic environment is relatively well known by humans, there is still a need to be able to communicate effectively and robustly in this environment as well as on land. Experience has proven that underwater communication is very restrictive and less stable. In order to better understand the challenges of this exercise, it is appropriate to study a state of the art of the technology, including the other aspects that we want to highlight. In this section, we will present an overview of underwater wireless sensor networks; the transport protocol concerned and related works.

2.1-Underwater Wireless Sensor Network overview

Underwater Wireless Sensor Network (UWSN) consists of a number of sensor nodes, stationary or mobile, connected wirelessly via electromagnetic or optical communication modules deployed to monitor various events of interest collaboratively over a given area. They can be used in challenging places where it is inconvenient for a human to be present. In [7] E. Felemban claims that sensors on the devices extract physical information from the environment, such as temperature through a temperature sensor, pressure through a barometer, noise through a microphone, and even an image through a camera or thermal camera. The collected data then is sent over to the control command for further processing. Frequency and data rate are mainly low in underwater communication with a set of nodes transmitting their data to a buoyant gateway that relays the data to nearest coastal monitoring and control station [8].

According to C. Peach and A. Yarali in [9], many characteristics of these networks make their construction challenging, and difficult to implement. These characteristics include limited bandwidth capacity, high error probability, large propagation delays, fouling, corrosion, high bit rate errors, limited battery power, and temporary losses of connectivity. However, as an advantage, sensor nodes and autonomous underwater vehicles must have the capability to self-configure themselves and be able to coordinate their configurations, location information, and movement information with each other.

2.1.1- Structure of Underwater-Sensor

A node is made up of four main parts. *Figure 2.1* gives an illustration of them:

- A power unit, consisting of a battery and a number of DC/AC converters,
- A processing unit, which usually consists of a small processor and memory,
- Physical sensors and

- The transceiver circuit that is formed by a transmitter and a receiver system

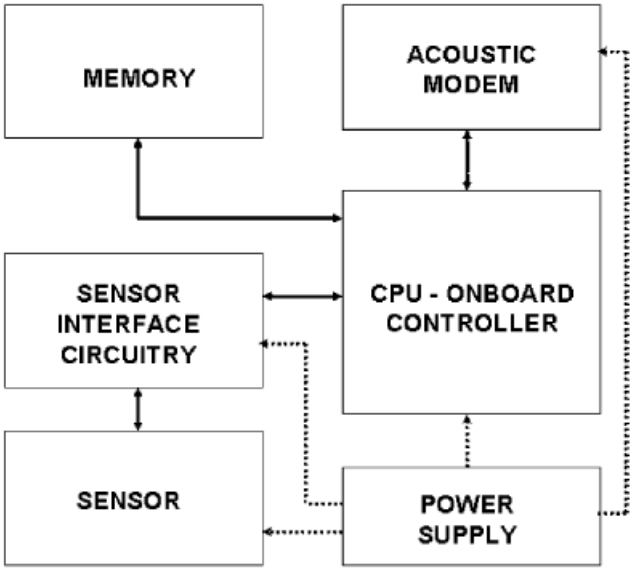


Figure 2.1- Internal architecture of an underwater sensor node [1]

Basically, the controller/CPU is interfaced with an oceanographic instrument or sensor through a sensor interface circuitry. It receives data from the sensor and it can store it in the onboard memory, process it, and send it to other network devices by controlling the acoustic modem. The electronics are usually mounted on a frame which is protected by a PVC (PolyVinyl Chloride) housing [1].

2.1.2- Underwater channel characteristics

Typical physical carriers for underwater communication signals are radio frequency (RF) electromagnetic waves, optical waves, and acoustic waves. Among the three types of waves, acoustic waves are used as the primary carrier for underwater wireless communication systems due to the relatively low absorption in underwater environments. This fact explains an increase of the literature related to this technique of retransmission whose characteristics are the following ones:

Transmission or path loss: Attenuation and geometric spreading are the main concern of the transmission loss.

- The attenuation is mainly provoked by absorption due to the conversion of acoustic energy into heat and increases with distance and frequency [10]. *Figure 2.2* shows the acoustic attenuation with varying frequency and distance for a short range shallow water UW-A (Underwater-Acoustic) channel, according to the propagation model. The attenuation is also caused by scattering and reverberation (on rough

ocean surface and bottom), refraction, and dispersion (due to the displacement of the reflection point caused by the wind on the surface). Water depth plays a key role in determining the attenuation as stated in [3].

- Geometric spreading: refers to the spreading of sound energy as a result of the expansion of the wave fronts. It increases with the propagation distance and is independent of frequency. [3] Outlines the two common kinds of geometric spreading: spherical (Omni-directional point source), which characterizes deep water communications, and cylindrical (horizontal radiation only), which characterizes shallow water communications. *Figure 2.2* illustrates the impact of both distance and frequency over the path loss.

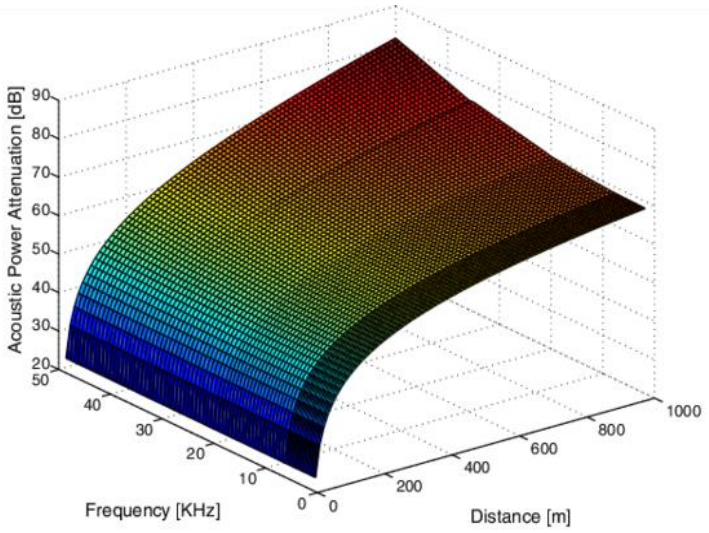


Figure 2.2- Path loss of short range shallow UW-A channels vs distance and frequency in band 1–50 kHz [1].

Noise: It can be classified as *man-made noise* and *ambient noise*. The former is mainly caused by machinery noise (pumps, reduction gears, power plants), and shipping activity (hull fouling, animal life on hull, cavitations), while the latter is related to hydrodynamics (movement of water including tides, current, storms, the wind, and rain), and to seismic and biological phenomena [1] [3] [5] [10]. In [11], the author claims that the modeling of the ambient noise related to four sources:turbulence in the water, breaking waves, thermal noise and shipping can be approximated by using the empirical formulas (in dB) where *f* refers to the frequency in kHz:

$$10\log N_t(f) = 17 - 30\log f \quad (1)$$

$$10\log N_s(f) = 40 + 20(s-0.5) + 26\log f - 60\log(f + 0.03) \quad (2)$$

$$10\log N_w(f) = 50 + 7.5w^{1/2} + 20\log f - 40\log(f + 0.4) \quad (3)$$

$$10\log N_{th}(f) = -15 + 20\log f \quad (4)$$

Where *t, s, w, th stand for* turbulence, shipping, wind and thermal respectively

Multipath: Multipath propagation can severely damage the acoustic signal, as it generates inter-symbol interference (ISI) [10]. The two main reasons inducing the multipath effect in underwater environments are discussed in [11] which are sound reflections at the surface, bottom or other objects*(cf. figure 2.3.a)*in the water and the varying sound speed*(cf. figure 2.3.b).*

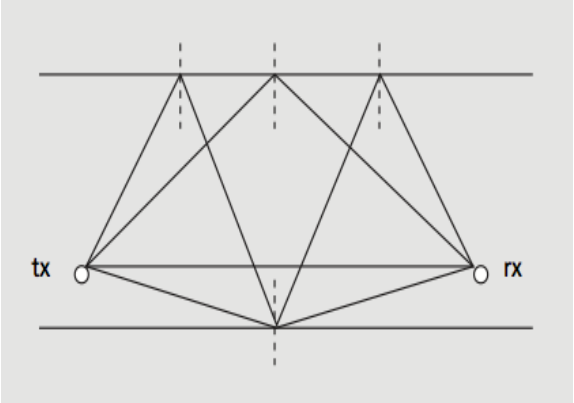


Figure 2.3.a- Multipath due to reflection on surface and bottom [11]

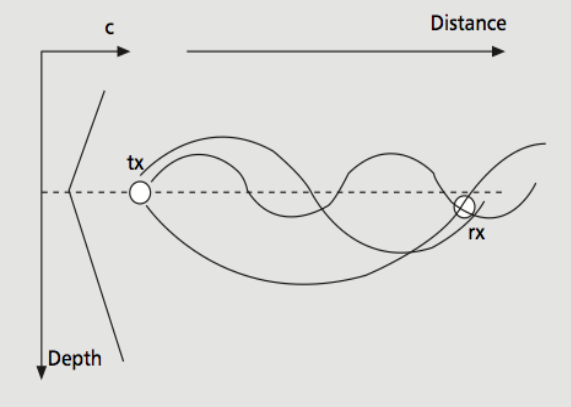


Figure 2.3.b- Multipath due to variating sound speed [11]

High delay and delay variance: The propagation speed in the UW-A channel is five orders of magnitude lower than in the radio channel. This large propagation delay (0.67s/km) and its variance can reduce the system throughput [1]. The underwater acoustic propagation speed can be expressed empirically as in [10]:

$$\mathbf{C\ (z,S,t) = 1449.05 + 45.7}t \mathbf{- 5.21}t^2 \mathbf{+ 0.23}t^3 \mathbf{+ (1.333 - 0.126}t \mathbf{+ 0.0009}t^2\mathbf{)*(S-35)+16.3z +0.18}z^2\mathbf{(5)}$$

Where **t** = 0.1×T, **T** represents the temperature in ^{o}C, **S** is the salinity in ppt, and **z** is the depth in km. The propagation speed varies between (1450 m/s -1540m/s).

Doppler spread: The Doppler frequency spread causes degradation in the performance of digital communications and generates two effects: a simple frequency translation and a continuous spreading of frequencies, which constitutes a non-shifted signal. The former is easily compensated at the receiver when the effect of the latter is harder to be compensated for [6]. Assuming a channel that has a Doppler spread with bandwidth **B** and a signal has symbol duration **T**, and then there are approximately **BT** uncorrelated samples of its complex envelope. When **BT** is much less than unity, the channel is said

to be underspread and the effects of the Doppler fading can be ignored, while, if greater than unity, it is said to be overspread [3].

2.1.3 Underwater communication architectures

Static two-dimensional, static three-dimensional and three-dimensional with AUVs (Autonomous Underwater Vehicles) are the three kinds of architecture that are discussed in [1] [3] [9] [12].

2.1.3.1 Static two-dimensional architecture

This network consists of sensor nodes that are anchored to the bottom of the ocean as shown in *figure 2.4.a*. Nodes are interconnected to underwater sinks [3] [9] [12] or underwater gateways [1]. These latter are responsible for relaying data from the ocean bottom network to surface station. Uw-gateways are equipped with two acoustic transceivers, namely horizontal (to communicate with the sensor nodes in order to send commands and configuration data to the sensors, collect monitored data) and a vertical (to relay data to a surface station) [1]. In [12] authors stated that vertical transceivers must be long range transceivers for deep water applications as the ocean can be as deep as 10 km. The surface station is equipped with an acoustic transceiver that is able to handle multiple parallel communications with the deployed underwater sinks. It is also endowed with a long range RF (Radio Frequency) and/or satellite transmitter to communicate with the onshore sink (Os-sink) or to a surface sink (s-sink).

2.1.3.2 Static three-dimensional architecture

Static three-dimensional underwater networks are used to detect and observe phenomena that cannot be adequately observed by means of ocean bottom sensor nodes, i.e., to perform cooperative sampling of the 3D ocean environment. In three-dimensional underwater networks, sensor nodes float at different depths to observe a phenomenon. As illustrated in *figure 2.4.b*, each sensor is anchored to the ocean bottom and equipped with a floating buoy that can be inflated by a pump. The buoy pushes the sensor towards the ocean surface. The depth of the sensor can then be regulated by adjusting the length of the wire that connects the sensor to the anchor, by means of an electronically controlled engine that resides on the sensor [1]. As outlined in [12] such architecture raises many challenges that need to be solved in order to enable 3D monitoring:

- Sensing coverage: Sensors should collaboratively regulate their depth in order to achieve full column coverage, according to their sensing ranges. Hence, it must be possible to obtain sampling of the desired phenomenon at all depths.
- Communication coverage: Since in 3D underwater networks there is no notion of Uw-sink, sensors should be able to relay information to the surface station via multi-hop paths. Thus, network devices should coordinate their depths such a way that the network topology is always connected, i.e., at least one path from every sensor to the surface station always exists.

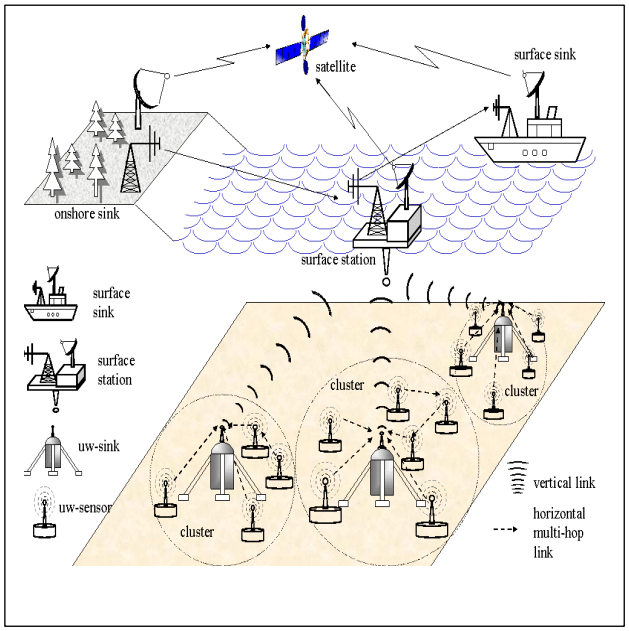


Figure 2.4.a Static Architecture for 2D underwater sensor networks [3]

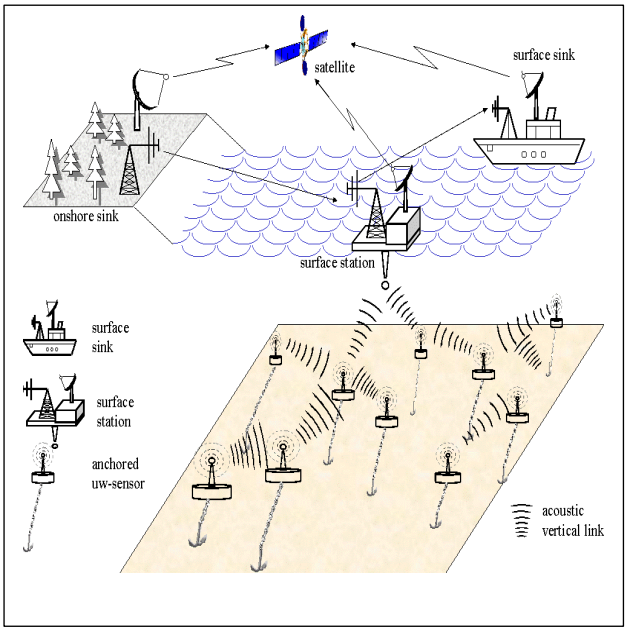


Figure 2.4.b Static Architecture for 3D underwater sensor networks [3]

2.1.3.3 Three-dimensional with autonomous AUVs

It consists of lots of static sensors together with some autonomous underwater vehicles (AUVs), as depicted in *figure 2.4.c*. AUVs play a key role for additional support in data harvesting. AUVs could be considered as super nodes, which have more energy, can move independently, and it could be a router between fixed sensors, or a manager for network reconfiguration, or even a normal sensor. In [1] authors propose a specialized architecture for UWSNs to provide energy efficient and robust architecture.

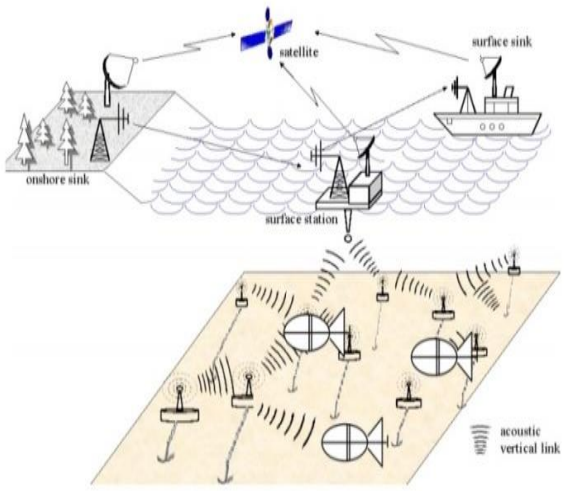


Figure 2.4.c Three-dimensional with AUV [13]

2.1.4 Layered model of UWSN

2.1.4.1 Physical layer

The function of the physical layer is to create a virtual link for transmitting a sequence of logical information between a pair of nodes. The transmitter converts bit streams into a physical signal that is propagated through the physical layer. On the other side, the receiver should be able to reverse the process and provide the original bit stream to the upper communication layers. One of the main issues tackled by the physical layer in a UWSN is the modulation. In [14], authors state the two kinds of modulation techniques discussed below:

Non-coherent modulation: Non-coherent signaling is commonly used to relay data and to transmit command-and-control information at low bitrates. Frequency-Hopped M-ary Frequency-Shift-Key (FH-MFSK) modulation is widely accepted as a reliable albeit inefficient choice for such modems. FSK (Frequency-Shift-Key) modulation uses distinct tonal or frequency-modulated pulses to map digital information. Frequency-hopping is a spread-spectrum technique, which alternates portions of the frequency band over fixed periods of time, so that successive FSK-modulated symbols use a different portion of the frequency spectrum, thus minimizing Inter-Symbol-Interference (ISI) caused by reverberation [15]. The main advantages of these modulations are their reliability and simplicity, so modems do not need high-resource processors with higher power consumption in [14]. Although non-coherent modulation schemes are characterized by high power efficiency, their low bandwidth efficiency makes them unsuitable for high data-rate multi-user networks [12]. Hence,

coherent modulation techniques have been developed for long-range, high-throughput systems. The evolution of this modulation technique is shown by *table 2.1*.

Table 2.1 Evolution of data rates for non- coherent modulation techniques [10]

Principal investigator	Data Rate [kbit/s]	Band [kHz] [kHz]	Bandwidth Efficiency	Range [km]	BER
Catipovic (1984)	1.2	5	0.4	3s	10^{-2}
Freitag a (1990)	2.5	20	0.13	3.7d	10^{-4}
Freitag a (1991)	0.6	5	0.12	2.9d	10^{-3}
Mackelburg (1991)	1.5	10	0.13	2d	N/A
Scussel (1997)	0.6 – 2.4	5	0.47	10d - 5s	N/A

d and *s* stand for *deep* and *shallow* water respectively. N/A indicates the data was not available in the published reference.

Coherent modulation: In order to increase the spectral efficiency and communication range, different alternatives based on phase-coherent modulation techniques, such as phase-shift keying (PSK) and quadrature amplitude modulation (QAM) have been developed [14]. The goal of these methods is to attain more than one bit transmitted per second per Hertz of occupied bandwidth. However, in order to face acoustic channel spread without significant spectrum efficiency loss, incoming signals should be equalized according to the channel response and decoded later. By using these techniques, modem complexity and power consumption increase [14]. *Table 2.2* shows the progress achieved in this domain.

Table 2.2 Evolution of data rates for coherent modulation techniques [10]

Principal investigator	Modulation Method	Data Rate [kbit/s]	Band [kHz] [km]	Range	BER
Suzuki (1989)	4, 8-PSK	20 – 30	10 / 25	3.5d	10^{-4}
Kaya (1989)	16-QAM	500	125 / 1000	0.06d	10^{-7}
Stojanovic (1993)	4, 8-PSK, 8-QAM	0.6 - 3.0	0.7 - 1.4	28 - 120s	10^{-2}
Labat (1994)	QPSK	6	3 / 60	4d	N/A
Capellano (1997)	BPSK	0.2	0.2 / 7	50d	10^{-4}
Freitag (1998)	QPSK	1.67 - 6.7	2 – 10	4.0s, 2.0s	N/A
Kojima (2002)	4, 8-PSK, 16-QAM	46, 96,128	40	0.03d	10^{-4}
Pelekanakis (2003)	8-PSK, 16, 32, 64-QAM	75, 100, 125, 150	60 – 90	0.01d	~0
Ochi (2010)	QPSK, 8-PSK	80, 120	80	0.84d, 0.62d	~0

d and *s* stand for *deep* and *shallow* water respectively. N/A indicates the data was not available in the published reference.

2.1.4.2 Data Link Control

The DLC (Data Link Control) is responsible for converting the unreliable bit pipe of the physical layer into a higher level error-free link. For this purpose DLC employs two mechanisms [16]:

- Framing: achieved by adding header information which consists of synchronization preamble, source and destination addresses at the beginning of the information sequence, and cyclic redundancy check (CRC) bits, at the end.
- Error correction control

2.1.4.3 Medium Access Control layer

Medium Access Control (MAC) protocols manage access to the communication medium and aim to avoid transmission collisions. However, they also have to deal with other factors, such as energy efficiency, scalability, and latency [14]. Medium access techniques can be broadly classified into two main categories [10]:

- Schedule-based, such as frequency-division multiple access (FDMA) and time-division multiple access (TDMA)
- Random-access based, such as ALOHA and carrier-sense multiple access (CSMA).

Moreover, CDMA (code-division multiple access)-based MAC protocols can be used in both scheduled and random-access based environments and possibly improves the system performance by allowing simultaneous code-division transmissions from multiple stations.

Due to the narrow bandwidth in UW-A channels and the vulnerability of limited band systems to fading and multipath, together with the often distributed nature of control in underwater networks, FDMA is rarely used. Pure TDMA schemes have also been proposed. However, TDMA shows limited channel utilization efficiency in large-scale networks because of the long time guards and/or heavy signaling requirements in UW-A links. Therefore, current underwater MAC solutions are for the most part based on random access schemes such as ALOHA, CSMA or CDMA. *Table 2.3* illustrates some pros and cons of each category of MAC protocol for underwater communications.

Table 2.3 Classification of MAC protocols in underwater communications [10]

	Pros	Cons
FDMA-based	Multiple users access simultaneously	Narrow bandwidth in UW-A channels and channels and vulnerability of Limited band system
TDMA-based	Avoiding collisions	Limited channel utilization efficiency in large-scale networks.
ALOHA-based	Easy to implement	Pure ALOHA has limited channel utilization
CSMA-based	Prevents collisions with ongoing Transmission	Channel may be sensed idle while a transmission is ongoing
CDMA-based	Robust to frequency-selective fading caused by underwater multipaths	Near-far problem reduces the performance

2.1.4.4 Network layer

The network layer is in charge of determining the path between a source (the sensor that samples a physical phenomenon) and a destination node (usually the surface station). In general, while many impairments of the underwater channel are adequately addressed at the physical and data link layers, some other characteristics, such as the extremely long propagation delays, are better addressed at the network layer [3]. There are two main methods to do this [11] :

- *Flooding techniques*: known as the simplest form of forwarding data where the source node broadcasts its packets to all its neighbors who again broadcast to its neighbors and so on. The packets may have a certain Time-To-Live (TTL) and are discarded if this limit is exceeded. This prevents the packets from living eternally in the network.
- *Routing:* Routing defines which way a packet shall take in the network to get to the destination using the best path based on metrics. One kind of metric can be hop-count and the best path will be the path with fewest hops, while other metrics can be bandwidth, delay, link quality or different combinations. The existing routing protocols are usually divided into three categories, namely *proactive*, *reactive* and *geographical* routing protocols:
  - *Proactive protocols*: attempt to minimize the message latency induced by route discovery, by maintaining up-to-date routing information at all times from each node to every other node. This is obtained by broadcasting control packets that

contain routing table information (e.g., distance vectors) [10] [12]. Nevertheless [11] notices that this results not only large amounts of overhead and information packets but also require some processing cycles on the nodes calculating routing tables. Proactive routing is best suited for static networks not having to update routing tables too often. Examples of proactive routing protocol that exists in WSN (Wireless Sensor Network) today are OLSR (Optimized Link State Routing) and DSDV (Destination-Sequenced Distance Vector).

- *Reactive protocols*: Reactive routing is a technique where a node does not store routing information before they need it but request a route whenever it has data to send. If a node (S) has data to send to another node (D) through a network, it starts by flooding a route request (RREQ) into the network. As the RREQ travels through the network, it stores the path it is sent. When the destined node receives the RREQ it waits for some time to see if more RREQ arrives with a different or better route. When the waiting timer expires, the node responds to the RREQ by sending a route reply (RREP). The RREP packet contains the best path from S -> D. Examples of reactive routing protocol that exists in WSN (Wireless Sensor Network) today are AODV (Ad hoc On-demand Distance Vector) and DSR (Dynamic Source Routing).
- *Geographical routing protocol:* is based on calculating the best route using the shortest geographic distance. One example used in terrestrial networks is the protocol GPSR (Greedy Perimeter Stateless Routing) that uses a greedy algorithm to find the shortest path to the destination. Here the nodes forward the packet to its one-hop neighbor who is closest to the destination. In the geographic routing protocols, the nodes have to know about their positions, and the most common way is to use GPS (Global Positioning System). In the underwater environment the GPS signal is quickly absorbed and is not suitable, but as alternatives to GPS, commercial underwater positioning systems based on acoustical transducers are available. Another way to enabling geographical routing is the use of nodes mounted on the bottom, pre-programmed with their position while the AUVs have some sort of navigation system for multiple purposes. *Table 2.4* illustrates some pros and cons of each routing protocol in underwater.

Table 2.4 Classification of routing protocols in underwater communications [10]

	Pros	Cons
Proactive	Routing protocol always tries to keep its routing data up-to-date	Scalability is a major issue
Reactive	Route is only determined when actually Needed	A higher latency is amplified by the slow propagation of acoustic signals
Geographical	Very promising for their scalability and localized signaling	GPS radio receivers do not work underwater

Figure 2.5 below illustrates the general classification of routing protocols developed for the UWSN. However, we will retain the DSDV protocol for our investigation

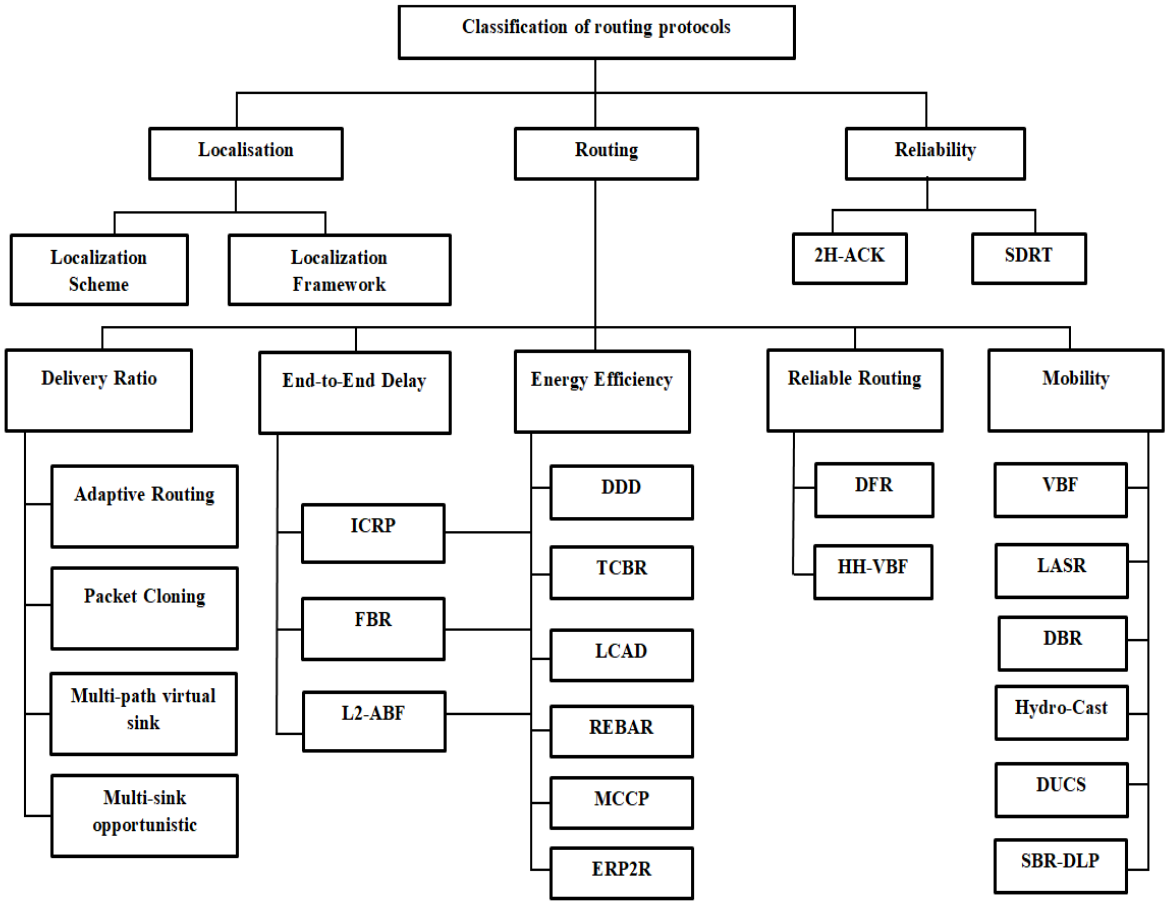


Figure 2.5 Knowledge based classification of routing protocols [17]

2.1.4.5 Transport layer

A transport layer protocol is needed in UWSN to achieve reliable transport of event features, and to perform flow control and congestion control. In [18] D. Pompili claims that existing TCP based on both window-based and rate-based mechanisms are unsuited for the underwaterenvironment. The first one relies on an accurate estimate of the Round Trip Time (RTT) which is long in the underwater environment, thus affect the throughput of most TCP implementations. Furthermore, the variability of the underwater RTT would make it hard to effectively set the timeout of the window-based mechanism. In the latter, the long and variable RTT can cause instability in the feedback control which enables flow control in this case. In this scope, Reeta Mishra has outlined the main challenges that the transport should deal with to overcome the unique conditions of underwater environment [19]:

- Flow control strategies to reduce high delay as well as delay variance of the control messages, efficient mechanisms to find the cause of packet loss and reliable network,
- To create solutions for handling the effect of connectivity losses caused by shadow zones, Flow control in data transmissions to avoid that network devices with limited memory are overpowered,
- Congestion control to prevent the network congestion.

To these may be added the proposals made by D. Pompili and Ian F. Akyildiz in [20] namely:

- Correctly handle shadow zones by predicting losses of connectivity and also interfacing with the routing layer.
- Minimize energy consumption by using the selective ACKnowledgment paradigm (SACK), which also helps preserve capacity on the reverse path.
- Rely on the rate-based transmission of data as this approach enables nodes to have a flexible control over the rates.
- Properly handle out-of-sequence packet forwarding.
- Timely react to local traffic impairments by relying on intermediate nodes so as to accelerate the response time in case of congestion.
- Leverage information from lower layers to predict, and then react to, losses of connectivity or partial packet losses.
- Be seamlessly integrated with hop-by-hop reliability mechanisms so as to locally recover packet losses without triggering costly end-to-end retransmission mechanisms

(that cannot be replaced totally by local recovery schemes, however, because hop-by-hop reliability does not guarantee end-to-end reliability).

Several solutions have been proposed to address the transport layer problems in UWSN and discussed below:

- *SDRT (Segmented Data Reliable Transfer):* A hybrid approach of ARQ and FEC, is effective in reducing the total number of transmitted packets significantly, improve channel utilization, and simplify protocol management [1] [20] [21]. The basic idea of SDRT is the use of « Tornado codes » to recover error packets to reduce retransmissions and thus reduce energy consumption by the mean of window control mechanism [20].
- *RTS (Reliable Transport and Storage):* A distributed fountain coding scheme is designed to facilitate reliable data delivery and multiple acknowledgments are adopted to ensure the control message reliability [22]
- *FRT (Fast and Reliable Transport):* takes advantage of the FEC mechanism and utilizes link quality toward neighbors so as to provide reliable and fast delivery of packets. It achieves less latency than an existing protocol without an increase of transmission overhead [23].
- NCRF (Network Coding in Rateless Fashion): A method for network coding that relies on implicit acknowledgments to improve power consumption performance. It takes advantage of the broadcast nature of the channel for the nodes to get an implicit acknowledgment of previously transmitted packets [24].
- *NCIA (Network Coding with Implicit Acknowledgment):* The principle lies on sending coded packets in burst, which only need an acknowledgment packet [25].
- *ADELIN (ADaptive rELIable traNsport):* The method guarantees transmission reliability only by sending enough coded packets [25].
- *CHARQ* (Cooperative Hybrid ARQ) protocol in [26] combines cooperative ARQ with incremental redundancy-hybrid ARQ. This protocol utilizes multiple relays to increase the system throughput, but it is still inefficient as the relay sets often do not obtain enough packets from the sender in the high packet error rate channel and many reduplicative check packets are sent. In a word, the previous Hybrid ARQ (HARQ) protocols achieve low channel utilization and perform inefficiently. In addition, they lack an adaptive code rate estimation for dynamic underwater channel to ensure the network can work at its optimal achievable performance

- *VBF-NC (Vector-Based Forwarding Network Coding)*: transfers packets, coded by network coding, over relay node sets, which are established by VBF (Vector Based Forwarding) routing protocol. However, only the error correction function of network coding is used by VBF-NC, the most important inbeing of network coding, which cannot only improve the throughput of the network but also reduce transmission overhead, is not used by VBF-NC [27].
- *ESRT (Event to Sink Reliable Transport):* in [3] [28], ESRT is proposed to achieve reliable event detection with minimum energy expenditure. However the basic relies on the spatial correlation among event flows which may not be easily leveraged in underwater acoustic sensor networks. In fact, in terrestrial sensor networks nodes are densely deployed, and thus the physical readings of spatially close nodes may be correlated (spatial correlation). Conversely, underwater sensor nodes may be more expensive and complex devices and are usually more sparsely deployed. Hence, the correlation among sensor readings from different sensors may not be significant in UWSNs.

Despite a large number of these transport protocols discussed above, none of them are implemented in the network simulator tool chosen for our study. As a result, we will study in section 2.2.3 those available in order to provide a solution to the stated problem.

2.1.4.6 Application layer

Although the application layer protocol for UWSNs remains largely unexplored, the purpose of the application layer can be outlined as follows [3] [18]:

- Provide a network management protocol that makes hardware and software details of the lower layers transparent to management applications;
- Provide a language for querying the sensor network as a whole;
- Assign tasks and to advertise events and data.

However, we can differentiate several classes of application [18]:

- Delay-tolerant application: includes media streams and data that do not need to be delivered within strict delay bounds and with low or moderate bandwidth demand. And can be divided into three categories: loss-tolerant multimedia streams, loss-tolerant data, loss-sensitive data
- Delay-sensitive application: includes video and audio streams, or multi-level streams composed of video/audio and other scalar data that need to reach a

human/automated operator in real-time; or data from time-critical monitoring processes such as distributed control applications

2.1.5 Cross-Layer approach

Cross-layer approach has been intensively developed in recent years because it does not have restrictions as layer approach [29]. In a traditional layered architecture, each layer interacts only with the adjacent layers in the protocol stack through well-defined interfaces. Although strictly layered architectures have served well the development of wired networks, they are known to be less than ideally suited for energy constrained wireless applications including UWSN [10]. In [30] authors argue that information in cross-layer architecture is exchanged between non-adjacent layers of the protocol stack as illustrated below in *figure 2.6*, typically using a broader and more open data format, and the end-to-end performance is optimized by adapting to this information at each protocol layer. So, instead of optimizing the performance of individual layer, it treats optimization as a problem for the entire stack. *Figure 2.6* illustrates a cross-layer design for the wireless protocol stack.

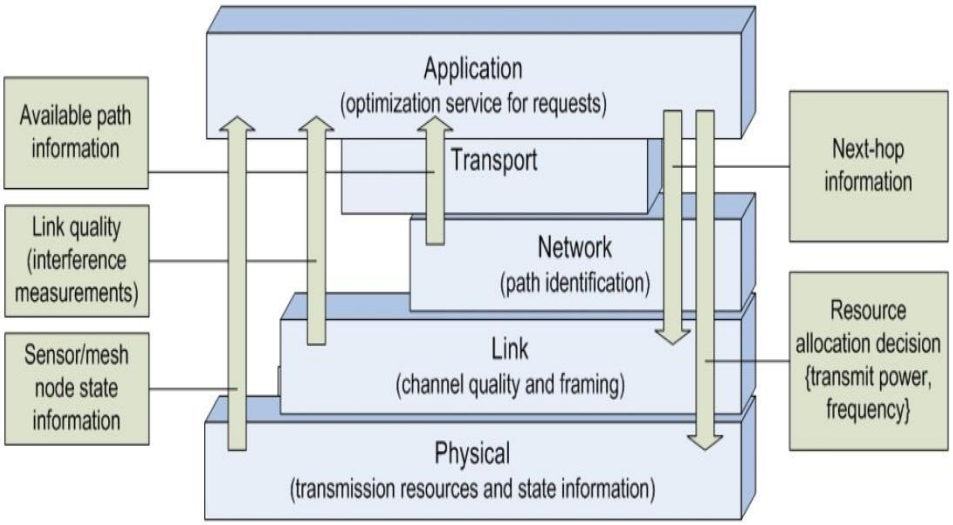


Figure 2.6 Cross-layer Design to enable smart routing [31]

2.1.6 Challenges of UWSNs

A study carried out by P. Vahdani Amoli clearly identified challenges to be faced in underwater environment considering the architectural model (exception to the application layer) of this technology. It highlights [32]:

- Physical implementation limitations:address the fundamental issue of loss or attenuation occurring in transmission, due to surface-bottom reflection, refraction

sound speed changes in depth and random changes caused by slow variations in the propagation medium.

- Medium access control and resource sharing: addresses the unique characteristics of the channel which are long delays, frequency-dependent attenuation, and the relatively long arrival of acoustic, plus bandwidth constraints of acoustic hardware that signals should be considered.
- Routing: it focuses on multi-hop routing efficiency, especially in the long-term mobile transmission which is a challenging aspect this communication. Also addressing localization of mobile sensor nodes as a crucial issue for underwater communication
- Reliable data transfer: evoking TCP's performance which may face problem due to channel errors and propagation delay.

To these may be added those largely outlined in [1] [3] [5] [6] [13] [20] [28] [33] [34] [35] [36] as follows:

- Scalability and self-organization
- The available bandwidth is severely limited and depends on the transmission distance
- Underwater sensors are expensive in terms of equipment, deployment, and maintenance.
- The memory limited by the capacity of the on-board storage device.
- Routing: due to high movement of nodes in water current.
- Range: usually used in vast ocean areas.
- Variation in sound speed due to temperature, pressure, and salinity. Variations in any one of these factors alter the sound speed. It may introduce error in distance estimation and accuracy of the scheme may decrease
- The underwater channel is severely impaired, especially due to multipath and fading problems
- Propagation delay in underwater is five orders of magnitude higher than in radio frequency (RF) terrestrial channels and extremely variable
- High bit error rates and temporary losses of connectivity (shadow zones) can be experienced, due to the extreme characteristics of the underwater channel
- Battery power is limited and usually, batteries cannot be recharged, also because solar energy cannot be exploited

- Underwater sensors are prone to failures because of fouling and corrosion
- Underwater sensors are characterized by high cost because of a small relative number of suppliers (i.e., not much economy of scale)

2.2- Transport Control Protocol review

Control-based transport protocols are used to decrease congestion and reduce packet loss, to provide fairness in bandwidth allocation and to guarantee end-to-end reliability. Although traditional transmission control protocol (TCP) is not suitable in UWSN, it is necessary to discuss it in order to clearly perceive the principle of reliable data transmission which is based on sequence acknowledgment and control of congestion by flow control. And then, go through the probable use of its variants implementations in UWSN and see if we can come out with significant results. Some of these variants will be also discussed.

2.2.1 Basics of TCP

TCP is a four-layer transport protocol that uses the basic IP services to provide applications with an end-to-end and connection-oriented packet transport mechanism [4]. This means ensuring a connection establishment (*figure 2.8*) between transceivers and a closing connection while communication ends (*figure 2.9*). Providing reliability is based on the principle of sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte the source expects to receive. Bytes not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request transmission. TCP provides efficient flow control, which means that, when sending acknowledgments back to the source, the receiving TCP process indicates the highest sequence number it can receive without overflowing its internal buffers. TCP offers both operations full-duplex (TCP processes can send and receive at the same time) and multiplexing (numerous simultaneous upper-layer conversations can be multiplexed over a single connection) [37].

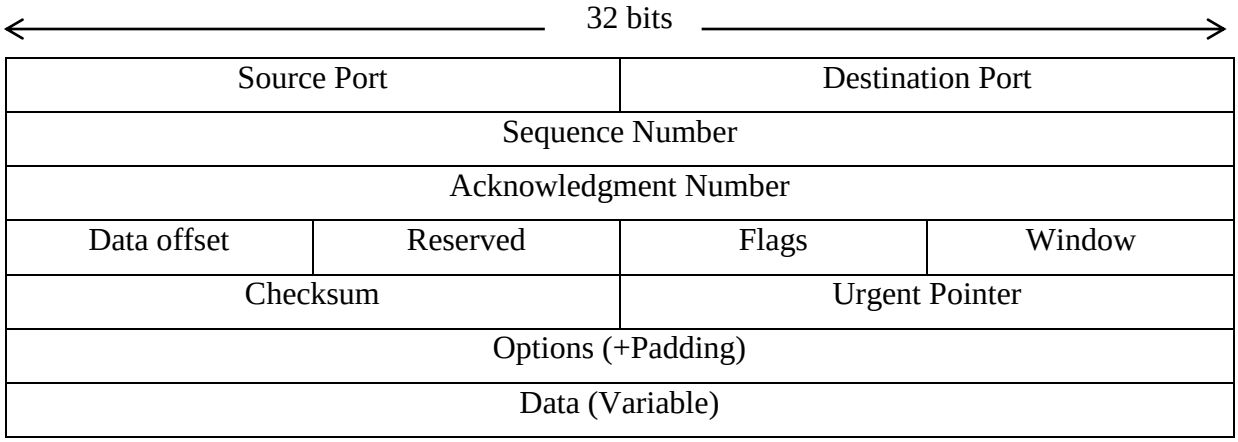


Figure 2.7 TCP segment format [37]

Figure 2.7 exhibits TCP segments fields which are summarized as follows:

- *Source Port* and *Destination Port*- identifies points at which upper-layer source and destination processes receive TCP services. Both are long of 16 bits
- *Sequence Number*- usually specifies the number assigned to the first byte of data in the current message. In the connection establishment phase, this field can be used to identify an initial sequence number to be used in an upcoming transmission. Long of 32 bits
- *Acknowledgment Number*- contains the sequence number of the next byte of data the sender of the packet expects to receive. Long of 32 bits
- *Data offset*- indicates the number of 32-bit words in the TCP header. Long of 4 bits
- *Reserved*- remains a reserve for future use (6 bits).
- *Flags*- carry a variety of control information, including the SYN and ACK used for connection establishment, and the FIN bit used for connection termination (6 bits).
- *Window*- specifies the size of the sender's receiving window (buffer space available for incoming data). Long of 16 bits
- *Checksum*- Indicates whether the header was damaged in transit (16 bits)
- *Urgent Pointer*- points to the first urgent data in the packet (16 bits)
- *Options* specifiy various TCP options
- *Data* contain upper-layer information

2.2.2 Mechanisms

2.2.2.1 Passive and active opens

TCP is a connection oriented protocol that requires before TCP traffic goes through, the application program at both ends must agree that the connection is desired. To do so, the application program on one end performs a passive open function by contacting its operating system and indicating that it will accept an incoming connection. At that time, the operating system assigns a TCP port number for its end of the connection. The application program at the other end must then contact its operating system using an active open request to establish a connection. The two TCP software modules communicate to establish and verify a connection. Once a connection has been created, the application program can begin to pass data; the TCP software modules at each end exchanges messages that guarantee reliable delivery [38].

2.2.2.2 TCP connection establishment

Connection establishment is performed by using a « three-way handshake » mechanism. The mechanism synchronizes both ends of a connection by allowing both sides to agree upon initial sequence numbers. This mechanism guarantees that both sides are ready to transmit and know that the other side is ready to transmit as well. This is necessary so that packets are not transmitted or retransmitted during session establishment or after session termination. Each host randomly chooses a sequence number used to track bytes within the stream it is sending or receiving [37]. Then, three-way handshake proceeds as in *figure 2.8* below.

2.2.2.3 TCP connection closing

When an application program tells TCP that it has no more data to send, TCP will close the connection in one direction. To close its half of connection, the sending TCP finishes transmitting the remaining data, waits for the receiver to acknowledge it, and then sends a segment with the FIN bit set. The receiving TCP acknowledges the FIN segment and informs the application on its ends that no more data is available. Once a connection has been closed in a given direction, TCP refuses to accept more data for that direction. Meanwhile, data can continue to flow in the opposite direction until the sender closes it. Nevertheless, acknowledgments continue to flow back to the sender even after a connection has been closed. When both directions have been closed, the TCP software at each point deletes its record of the connection. The illustration is provided with *figure 2.9.*

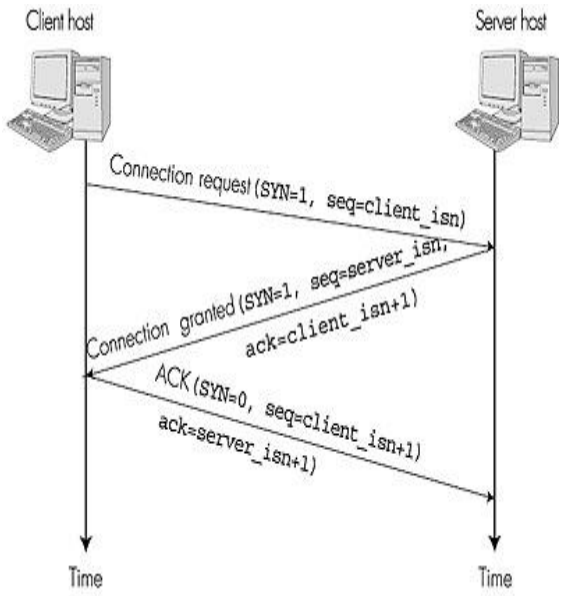


Figure 2.8 Illustration of a Three-way handshake mechanism [39]

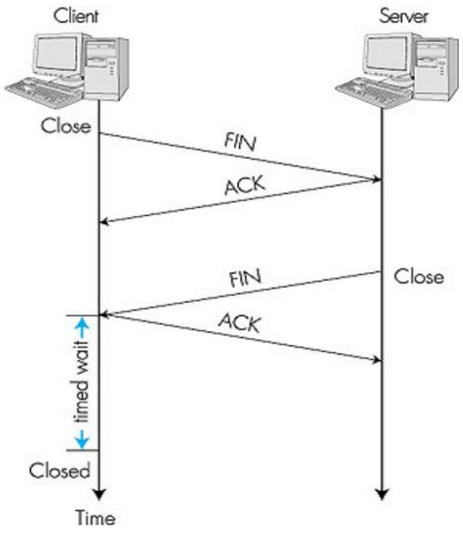


Figure 2.9 Closing the TCP connection [39]

2.2.2.4 Acknowledgements

Acknowledgment is a specific sequence value by which receiver sends feedback to the sender to notify the reception of segments of the stream. Basically, receiver collects data octets from arriving segments and reconstructs an exact copy of the stream being sent. Since segments travel in IP datagrams, lost and out-of-order delivery can occur; so that receiver uses the sequence number to reorder segments. Moreover, thereceiver always acknowledges the longest contiguous prefix of the stream that has been received correctly. Acknowledgment specifies the sequence number of the next octet that the receiver expects to receive [38].

2.2.2.5 Timeout and retransmission

Whenever a TCP sender transmits a segment, it starts a timer and waits for an acknowledgment. If the timer expires before data in the segment has been acknowledged, TCP assumes the segment was lost or corrupted and then, retransmits it [38].

2.2.2.6 Slow-start and Congestion avoidance

The slow start and congestion avoidance algorithms are used by a TCP sender to control the amount of outstanding data (data sent but not acknowledged yet), being

injected into the network. In fact, the minimum of the two state variable CWND (Congestion Window is a sender-side limit on the amount of data the sender can transmit into the network before receiving an acknowledgment) and RWND (Receiver's advertised Window is a receiver-side limit on the amount of outstanding data) governs data transmission. Another state variable, SSTHRESH (Slow-Start THRESHold), is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission. Beginning transmission into a network with unknown conditions requires TCP to slowly probe the network to determine the available capacity, in order to avoid congesting the network with an inappropriately large burst of data. The slow start algorithm is used for this purpose at the beginning of a transfer, or after repairing loss detected by the retransmission timer. Slow start additionally serves to start the "ACK clock" used by the TCP sender to release data into the network in the slow start, congestion avoidance, and loss recovery algorithms. The slow start algorithm is used when CWND < SSTHRESH, while the congestion avoidance algorithm is used when CWND > SSTHRESH. When CWND and SSTHRESH are equal, the sender may use either slow start or congestion avoidance [40]. How do both work?

- During slow start, TCP increments CWND by at most SMSS (Sender Maximum Segment Size) bytes for each ACK received that cumulatively acknowledges new data. Slow start ends when cwnd exceeds SSTHRESH (or, optionally, when it reaches it, as noted above) or when congestion is observed. In [40], a recommendation about the CWND increase is given as follow: CWND += min (N, SMSS), where N is the number of previously unacknowledged bytes acknowledged in the incoming ACK. This provides robustness against misbehaving receivers that may attempt to induce a sender to artificially inflate CWND using a mechanism known as "ACK Division". ACK Division consists of a receiver sending multiple ACKs for a single TCP data segment, each acknowledging only a portion of its data. A TCP that increments CWND by SMSS for each such ACK will inappropriately inflate the amount of data injected into the network.
- During congestion avoidance, CWND is incremented by roughly 1 full- sized segment per round-trip time (RTT). Congestion avoidance continues until congestion is detected. The recommendation done by [40] is to count the number of bytes that have been acknowledged by ACKs for new data. When the number of bytes acknowledged reaches CWND, then CWND can be incremented by up to SMSS bytes. Note that during congestion avoidance, cwnd must not be increased by more than SMSS bytes

per RTT. This method both allows TCPs to increase CWND by one segment per RTT in the face of delayed ACKs and provides robustness against ACK Division attacks.

2.2.2.7 Fast Retransmit and Fast Recovery

The TCP sender uses the "fast retransmit" algorithm to detect and repair loss, based on incoming duplicate ACKs which may be caused either by dropped segments, either by the re-ordering of data segments by the network or by replication of ACK or data segments by the network. The fast retransmit algorithm uses the arrival of 3 duplicate ACKs as an indication that a segment has been lost and enters Fast recovery. After receiving 3 duplicate ACKs, TCP performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire. After the fast retransmit algorithm sends what appears to be the missing segment, the "fast recovery" algorithm governs the transmission of new data until a non-duplicate ACK arrives [40]. This discussion of TCP mechanisms is not exhaustive, it may extend to others such as those outlined in [37] and [38]: PAR (Positive Acknowledgement and Retransmission), TCP sliding window, TCP checksum computation, accurate measurement of round trip samples, TCP connection reset, TCP state machine, etc.

2.2.3 TCP variants

The review of TCP variants was motivated by the fact that none of the solutions discussed in section 2.1.4.5 is implemented in our simulator that will serve for our experiments. Since these TCP variants operate in an ad-hoc WSN (Wireless Sensor Network) environment, this may provide a basis for underwater environment operation.

2.2.3.1 Tahoe

Tahoe is the simplest among the five variants we are intended to discuss in this section. Early TCP implementations followed « go-back-n » model using cumulative positive acknowledgment and requiring a retransmit timer expiration to resend data lost during transport. As a result, these TCPs did little to minimize network congestion [41]. For these reasons, Tahoe was built with new algorithms and refinements to earlier implementations namely: slow-start, congestion avoidance, fast retransmit and a modification to the round-trip time estimator used to set retransmission timeout values as claim authors K. Fall and S. Floyd in [41]. Let see how slow-start process works with Tahoe: It suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called ‘slow-start’. The reason for this procedure is

that an initial burst might overwhelm the network and the connection might never get started. Slow start suggests that the sender set the congestion window to 1 and then for each ACK received it increase the CWND by 1. So in the first Round Trip Time (RTT) we send 1 packet, in the second we send 2 and in the third, we send 4. Thus we increase exponentially until we lose a packet which is a sign of congestion. When we encounter congestion we decrease our sending rate and we reduce congestion window to 1. And start over again [42]. In congestion avoidance, Tahoe uses AIMD (Additive Increase Multiplicative Decrease) and increments linearly until it encounters a packet loss. Thus it increases it window slowly as it approaches the bandwidth capacity. We will not go through others algorithms that have been discussed previously but just highlight shortcomings faced by implementing Tahoe. As stated in [42], the problem with Tahoe is that it takes a complete timeout interval to detect a packet loss. Also since it does not send immediate ACKs, it sends cumulative acknowledgments, therefore it follows a 'go-back-n' approach. Thus every time a packet is lost it waits for a timeout and the pipeline is empty. This offers a major cost in high bandwidth delay product links.

2.2.3.2 Reno

Reno retains the basic principle of Tahoe but modified the Fast Retransmit operation to include Fast Recovery to prevent the "pipe" from going empty after Fast Retransmit. Thereby, it avoids the need to enter slow-start to re-fill it after a single packet loss [41]. Fast Recovery is entered by a TCP sender after reaching a threshold of duplicate ACKs (generally set to three). Then the sender retransmits one packet and reduces its congestion window by one-half. However, Reno continues to have a deficiency: it does not work well in the event of multiple losses within a single sending window. [41] [42]. Another problem is that if the window is very small when the loss occurs then we would never receive enough duplicate acknowledgments for a fast retransmit and we would have to wait for a coarsely grained timeout. Thusit cannot effectively detect multiple packet losses.

2.2.3.3 Newreno

Newreno is a slight modification over TCP Reno. It is able to detect multiple packet losses and thus is much more efficient than Reno when multiple packet losses arise. The change concerns the sender's behavior during Fast Recovery when a partial ACK is received that acknowledges some but not all of the packets that were outstanding at the start of that Fast Recovery period. In Reno, partial ACKs take TCP

out of Fast Recovery by “deflating” the usable window back to the size of the congestion window. In Newreno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, Newreno can recover without a retransmission timeout, retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. Newreno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged [41] [43].

2.2.3.4 Sack

SACK (Selective ACKnowledgments) is an extension of TCP Reno and it works around the problems faced by TCP Reno and TCP New-Reno, namely detection of multiple lost packets, and retransmission of more than one lost packet per RTT and retains the slow-start and fast retransmits parts of Reno. As claimed by M. Mathis, J. Mahdavi, PSC S. Floyd in [44], a selective acknowledgment allows the data receiver to inform the sender about all segments that have arrived successfully, so the sender only needs to retransmit the segments that have actually been lost.

Each ACK has a block which describes which segments are being acknowledged. Thus the sender has a picture of which segments have been acknowledged and which are still outstanding. Whenever the sender enters fast recovery, it initializes a variable « *pipe* » which is an estimate of how much data is outstanding in the network, and it also set CWND to half the current size. Every time it receives an ACK it reduces the pipe by 1 and every time it retransmits a segment it increments it by 1. Whenever the pipe goes smaller than the CWD window, it checks which segments are not received and send them. If there are no such segments outstanding then it sends a new packet [42]. The sender ends Fast Recovery when a recovery acknowledgment is received acknowledging all data that was outstanding when Fast Recovery was entered. Another feature handling by TCP SACK in discussed in [41] and concerns the partial ACKs. As mentioned by Fall and Loyd, in partial ACK the sender decrements pipe by two packets rather than one, as follows. When Fast Retransmit is initiated, the pipe is effectively decremented by one for the packet that was assumed to have been dropped, and then incremented by one for the packet that was retransmitted. Thus, decrementing the pipe by two packets when the first partial ACK is received is in some sense “cheating”, as

that partial ACK only represents one packet having left the pipe. However, for any succeeding partial ACKs, the pipe was incremented when the retransmitted packet entered the pipe but was never decremented for the packet assumed to have been dropped. Thus, when the succeeding partial ACK arrives, it does, in fact, represent two packets that have left the pipe: the original packet (assumed to have been dropped), and the retransmitted packet. Because the sender decrements pipe by two packets rather than one for partial ACKs, the SACK sender never recovers more slowly than a Slow-Start. The biggest problem with SACK is that currently selective acknowledgments are not provided by the receiver to implement SACK, so we'll need to implement selective acknowledgment which is not a very easy task [42].

2.2.3.5 Vegas

Vegas is an enhancement of Reno, based on the proactive approach which is considered more efficient than reactive ones to deal with congestion [42]. The major changes are discussed and outlined as follows [45]:

- New retransmission mechanism
- Congestion avoidance mechanism
- Modified slow-start mechanism

New retransmission mechanism

This technique results in a more timely decision to retransmit a dropped segment. The principle is described L. Brakmo and L. Peterson as follows: Vegas keeps track by recording the system clock each time a segment is sent. When an ACK arrives, Vegas reads the clock again and computes the RTT using this time and the timestamp recorded for the relevant segment. It then uses this more accurate RTT estimate to decide to retransmit in the following two situations:

- When a duplicate ACK is received, Vegas checks to see if the difference between the current time and the timestamp recorded for the relevant segment is greater than the timeout value. If it is, then Vegas retransmits the segment without having to wait for n (3) duplicate ACKs.
- When a non-duplicate ACK is received, if it is the first or second one after a retransmission, Vegas again checks to see if the time interval since the segment was sent is larger than the timeout value. If it is, then Vegas retransmits the segment. This will catch any other segment that may have been lost previously to the retransmission without having to wait for a duplicate ACK.

The goal of the new retransmission mechanism is not just to reduce the time to detect lost packets from the third duplicate ACK to the first or second duplicate ACK—small savings—but to detect lost packets even though there may be no second or third duplicate ACK. The new mechanism is very successful at achieving this goal, as it further reduces the number of coarse-grain timeouts in Reno by more than half.

Congestion avoidance mechanism

This technique gives TCP the ability to anticipate congestion, and adjust its transmission rate accordingly. L. Brakmo and L. Peterson in [45] give the algorithm details as follows:

- First, Set a *BaseRTT* to be the RTT of a segment when the connection is not congested. Commonly the RTT of the first segment sent by the connection, before the router queues increase due to traffic generated by this connection. Then the expected throughput is given by***Expected= WindowSize/BaseRTT***, where WindowSize is the size of the current congestion window, assuming to be equal to the number of bytes in transit
- Second, Vegas calculates the current Actual sending rate by recording the sending time for a distinguished segment, recording how many bytes are transmitted between the time that segment is sent and its acknowledgment is received, computing the RTT for the distinguished segment when its acknowledgment arrives and dividing the number of bytes transmitted by the sample RTT. This calculation is done once per round-trip time.
- Third, Vegas compares Actual to Expected and adjusts the window accordingly. Let say, Diff= Expected - Actual. Also define two thresholds, $\alpha<\beta$, roughly corresponding to having too little and too much extra data in the network, respectively. When $Diff < \alpha$, Vegas increases the congestion window linearly during the next RTT, and when $Diff > \beta$, Vegas decreases the congestion window linearly during the next RTT. Vegas leaves the congestion window unchanged when $\alpha < Diff < \beta$.

Thus, the farther away the actual throughput gets from the expected throughput, the more congestion there is in the network, which implies that the sending rate should be reduced. The β threshold triggers this decrease. Otherwise, the αthreshold triggers the increase.

Modified slow-start algorithm

It aims to detect and avoid congestion during slow-start. TCP Vegas differs from the other algorithms during its slow-start phase. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion [42]. To be able to detect and avoid congestion during slow-start, Vegas allows exponential growth only every other RTT. In between, the congestion window stays fixed so a valid comparison of the expected and actual rates can be made. When the actual rate falls under the expected rate by the equivalent of one router buffer, Vegas changes from slow-start mode to linear increase/decrease mode [45].

2.3 Related works

This section provides some improvements of TCP performance in UWSN and WSN as well. This may constitute a basis for our work.

1) Authors propose in [46] a method which limits the maximum window size. As a result, this leads to increase the throughput. In this multi-hop architecture, the author used fixed nodes numbered from 0 (the transmitter) to N (the receiver) with a range of 70 m between nodes and a link bandwidth set at 150 kbps. They come out with two criteria: the effect of the number of hops and the effect of the maximum sending window limitation over Hybla. And then, he has compared the results between improved Hybla and those related to Newreno. It has been shown :

- Although Hybla presented better throughput than Newreno when the number of hops is between 2 and 5, it still requires an adequate setting of RTT0 to reduce the buffer overflow. But this represents a challenging issue because the RTT is not stable in UWSN environment.
- Hybla did not experience any significant improvement when applying the maximum sending window limitation. Contrariwise, Newreno throughput is improved in all the packet size cases.

2) In [47], authors provide a model in order to ensure reliable data deliveries from sensor nodes to surface sink. For this purpose, they propose an algorithm which determines the suitable data packet size for efficient data transfer. The solution proposed by authors is based on the cross-layer approach which extends the HbH-ACK (Hop-by-Hop ACKnowledgement) scheme to 2H-ACK scheme. This new scheme proposes that three nodes be involved in the transmission of one segment with 2 nodes retaining the copy of the relevant segment so that in the case of an unwanted event such as

destruction of the node, another copy is always available on the network. This method has shown advances, among other things, the improvement of packets lost from 92 (HbH-ACK case) to 24 (2H-ACK case) or +72.3% gain. A detail of the algorithm used to compute the optimum packet size is also provided in section 5.

3) In [48], authors propose CTCP (Collaborative Transport Control Protocol), a method to recover the packets discarded by a buffer overflow in the multi-hop data transmission. It is based on two levels of reliability. Level 1 where only two nodes are involved in the forwarding of the packet like HbH-ACK scheme; and level 2 resembling the 2H-ACK scheme discussed in [48] but with the difference that the transmitter erases the segment from its buffer after receiving a double ACK guaranteeing increased reliability detailed in section 4B. This solution shows performance improvement with the average delivery rate of 29.82%, 61.11%,and 98.15% respectively without transport protocol, with CTCP level-1 and CTCP level-2 situations.

This section was part of a state-of-the-art study of UWSN, TCP technologies, and the related works in this domain. It enabled us to highlight and retain the fundamental characteristics of the two technologies mentioned above. This will allow us to approach the next chapter with certain precautions or attitudes that would lead us to our goal. Among these attitudes we can enumerate:

- A rigor in the scenario definition (the segment size, sensor layout, application protocol, etc.)
- The choice of the TCP variant to be used
- Definition of metrics for the evaluation of TCP performance
- The taking into account of certain parameters which could have an impact on the behavior of TCP

3 U-Newreno to improve TCP performance in UWSN

3.1 Presentation of U-Newreno

UWSN communication is known to be conducive to packet loss during transmission while TCP is a protocol whose performance is very sensitive to packet loss. This is a challenge to use TCP as the transport protocol for such a communication. We propose U-Newreno (Underwater-Newreno) which is an enhancement of Newreno to improve the performance of TCP in a UWSN communication.

Indeed, TCP is a reliable data transmission protocol that relies on several mechanisms as we discussed earlier. By better understanding, its mechanismslead to a better understanding of the TCP working and intuitively anticipate changes or options to be set for a given environment. So how has Newreno been improved?

Two methods have caught our attention: congestion control and RTT timeout. These address two challenges in the UWSN communication presented above in section 2.1.6 namely a limited bandwidth, long and variable delay propagation. The improvement is to vary the values of the size of the upper bound of the congestion window and the parameter "rtxcur_init". The values for these two parameters are between 1 and 20 with an increment by 1.Thus the efficient use of the bandwidth is addressed by the control of the upper limit of the sending window and the adaptation of the RTT timeout concerns the setting of "rtxcur_init".

We have added another approach to our solution. This consists in following the proposal of M. Allman, V. Paxson, ICSI, E. Blanton in [40]. Indeed, these are recommendations about the size of the upper limit of the initial window (IW) according to the sender maximum segment size (SMSS) defined as follows:

- If SMSS> 2190 bytes: IW = 2 * SMSS bytes and MUST NOT be more than 2 segments
- If (SMSS> 1095 bytes) and (SMSS <= 2190 bytes): IW = 3 * SMSS bytes and MUST NOT be more than 3 segments
- If SMSS <= 1095 bytes: IW = 4 * SMSS bytes and MUST NOT be more than 4 segments

For this reason we will use three different segment sizes that fit the above recommendations in our simulations. Taking into account these recommendations is motivated by the fact that firstly we do not master the behavior of TCP in the UWSN.

Second, we have no apprehension about a probable relation between the size of the segment and the performance of TCP.

The following parameters SMSS, initial window, upper bound of the congestion window, rtxcur_initare respectively named ‘packetSize_’, ‘windowInit_’, ‘window_’ and ‘rtxcur_init_’ in the simulator tool used. And they are located in ~ /aquasim/ns-2.30 /tcl/lib/ ns-default.tcl file. In addition, the RTT timeout algorithm is defined by the method ‘TcpAgent :: rtt_timeout ()’ contained in the TCP class (~/aquasim/ns-2.30/tcp/tcp.cc).This method is also conditioned by [67] which define how to compute the TCP retransmission timer. The results of our solution will be presented and evaluated in section 3.3.

3.2 Objectives of U-Newreno

The objectives refer to the expectations placed in U-Newreno. Those are:

- Better management of flow control through a suitable transition between slow-start and congestion avoidance phases.
- Avoiding abusive RTT timeout

3.2.1 Management of the flow control

Modern implementations of TCP contain four intertwined algorithms: slow-start, congestion avoidance, fast retransmit, and fast recovery.

Let's look at the first two algorithms listed above. In practice, these two mechanisms are used to manage the sending window size of the transmitter during a transmission. When a transmission begins, TCP uses the slow-start phase to probe the network and estimate its capabilities (bandwidth, bit error rate, RTT, etc.). For this reason, the behavior of TCP during this phase is important especially in a UWSN communication where the channel characteristics are critical. During the slow-start phase, the sending window grows exponentially which can lead fairly quickly to a large window size. And a large size of CWND translates into multiple segments simultaneously injected into the network and therefore a greater demand for bandwidth. This could increase the probability of packet loss because the bandwidth is limited in UWSN.

And on the other side, the congestion avoidance phase has flexibility in increasing the sending window. This is incremented by 1 per RTT.

Thus, by controlling the maximum limit of the size of the sending window, the idea is to allow TCP to evolve with a reasonable sending window. So that, the transition between the two phases, which is induced by crossing the SSTHRESH, occurs more or less

rapidly. This would allow TCP operation in congestion avoidance mode. As a result, it would enable an incrementation of 1 of the sending window. This would probably prevent the retransmission of several packets that could be lost in a large sending window. Therefore, it would allow TCP to rationally use the bandwidth of the channel.

3.2.2 Avoiding abusive RTT timeout

The default value of the RTT timeout is 3 sec when computing the TcpAgent :: rtt_timeout () method. Taking into account the channel characteristic regarding the propagation delay (known to be long and variable), we have no apprehension that the value of the default RTT timeout is appropriate for the UWSN. By tuning the parameter "rtxcur_init" which initializes the parameter "t_rtxcur", this induces the estimation of the RTT timeout.

By adapting the appropriate RTT timeout to a UWSN communication, the objective is to avoid retransmission timeout (RTO) occurring too early that would result in retransmissions. The idea of avoiding the occurrence of abusive RTOs would simply result in a reduction in packet retransmissions. This could be seen as an improvement in TCP performance.

3.3 Simulation and Performance evaluation

Undertaking sea trials or even shallow water would probably have been the best way to make experiments more realistic, and at the same time better understand the behavior of TCP in an underwater environment. But such experiments are very costly in terms of:

- Staff: it must be highly qualified
- Infrastructure: the need to have real equipment (sensor nodes, sink, computers, etc.)
- Software: the need to have adequate ones to record and process the collected data.

The fact is that we do not have this system. However, it would still be necessary to put this complex system in place to achieve our research goal. This is why simulation is the appropriate solution available at this stage of the research. It has the advantage of being able to carry out the tests on a single computer while respecting the requirements and makes it possible to get closer to the real world. To this end, a study of several simulation tools will be carried out in order to choose the one which best corresponds to our expectations. Then the evaluation of the results will be done. It will illustrate the progress achieved.

3.3.1 Simulation tools

3.3.1.1 GloMoSim

GloMoSim stands for GLObalMObile information system SIMulator and satellite network simulation environment developed at UCLA parallel computing laboratory [49]. It is a library-based sequential and parallel simulator for wireless networks, designed as a set of library modules, each of which simulates a specific wireless communication protocol in the protocol stack. The library has been developed using PARSEC (C-based parallel simulation language). New protocols and modules can be programmed and added to the library using this language. Implemented on both shared memory and distributed memory computers and can be executed using a variety of synchronization protocols, GloMoSim is extensible and composable [50]. Its design is provided by *figure 3.1*.

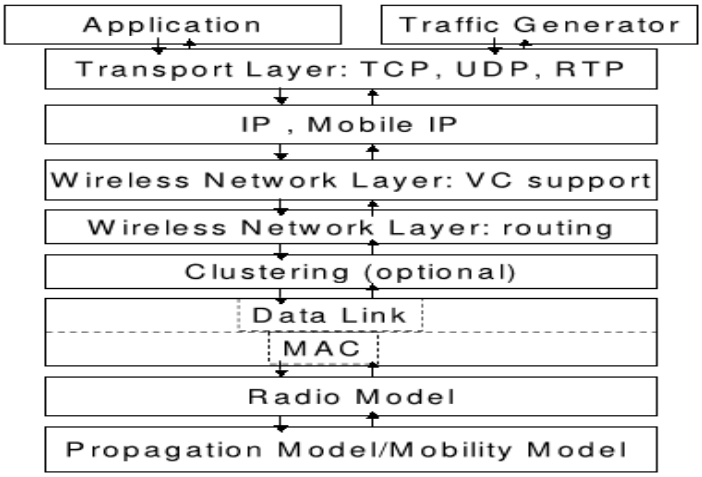


Figure 3.1 GloMosim Architecture [50]

3.3.1.2 UnderWater Simulator

UnderWater Simulator (UWSim) is primarily designed for Underwater Sensor Network. UWSim is mainly focusing on some specific handling scenarios like low-bandwidth, low frequency, high transmission and limited memory. It is based on component-based approaches slightly than a layer/protocol-based approach. [51] outlines the main UWSim features as follows :

- *A configurable environment*: the 3D-architecture is easily configured thanks to modeling software as Blender, 3D Studio Max, etc. In addition to the basic 3D structure, additional elements can be dynamically added, modified and removed from the main program.

- *Multiple robots support* consists of abstract classes that can be specialized for adding support for different vehicles and underwater manipulators.
- *Simulated sensors*: Twelve sensors are available for vehicles plus default position/velocity sensors available for vehicles and manipulator joints in the current version of UWSim (1.4). The available sensors are Camera, Range Camera, Range Sensor, Object picker, Pressure, DVL, Imu, GPS, Multibeam, Force, Structured light projector, Dredge.
- *Contact physics*: Physics simulation is also supported, although in a simple manner at this moment. This is done by integrating the physics engine Bullet with OSG through osgBullet. This allows simulating contacts and forces and automatically updating the scene accordingly. The different bodies collision shapes can be automatically generated from the 3D models. In order to have water physics, underwater vehicle dynamics must be used (already available in the underwater simulation stack)
- *Network interfaces*: All the different robots sensors and actuators can be interfaced with external software through the network. For this, It has been integrated the simulator into the Robot Operating System (ROS) that provides many facilities for communications and distributed computation. Through the network interfaces, it is possible to access/update any vehicle position or velocity, to move arm joints, or to access simulated sensors such as images generated by virtual cameras. However, the core software is independent of any middleware, so it is possible to interface the simulator with other middlewares, it is recommended to take ROSInterface.h as an example.
- *Widgets*: Widgets are small windows that can be placed on the main display in order to show specific data to the user. An abstract interface for the creation of custom widgets is included. It allows creating specialized classes for displaying user data depending on the specific application.

3.3.1.3 Network Simulator 2

Network Simulator v2 (NS2) is a discrete event simulator targeted at networking research. NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995, NS development was supported by DARPA (Defense Advanced Research Projects Agency) through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently, NS development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with

other researchers including ACIRI [52]. The simulator is written in C/C++ and is interfaced with TCL (Tool Command Language) /OTCL (Object-oriented extension of Tcl) [11] [53]. The distribution between the two languages is that the simulator kernel and the network modules are written in C/C++. Interfacing with the simulator is done with TCL/OTCL where the network will be initiated, the topology built and the different events in the simulation configured. With this distribution of languages, the compiled modules perform well, but there is no need to recompile whenever a change occurs in the topology [11]. However, NS2 cannot handle a simulation in the underwater environment because this environment requires specific components that are not built-in. But the interest to present it comes from the fact that the rest of simulation tools that will be discussed are NS2- based.

3.3.1.3.1 Tool Command Language and Object-oriented TCL

Tool Command Language (TCL) is most probably used for writing simulation code. An example in [55] is provided to show how to program in TCL.

OTCL is an extension of Tcl with object-oriented programming. It is dynamically extensible and built on Tcl syntax and concepts. The network simulator NS is actually a special version of an OTcl programming language interpreter [54] [55]. In [55] a basis is provided for learning about OTCL.

3.3.1.3.2 Trace file

A trace file (*cf. figure 3.2*) is used to collect data and processed to evaluate simulation results. With the aid of NS2 trace file, drop or arrival of packets that occurs in the queue or in a link are recorded [53].

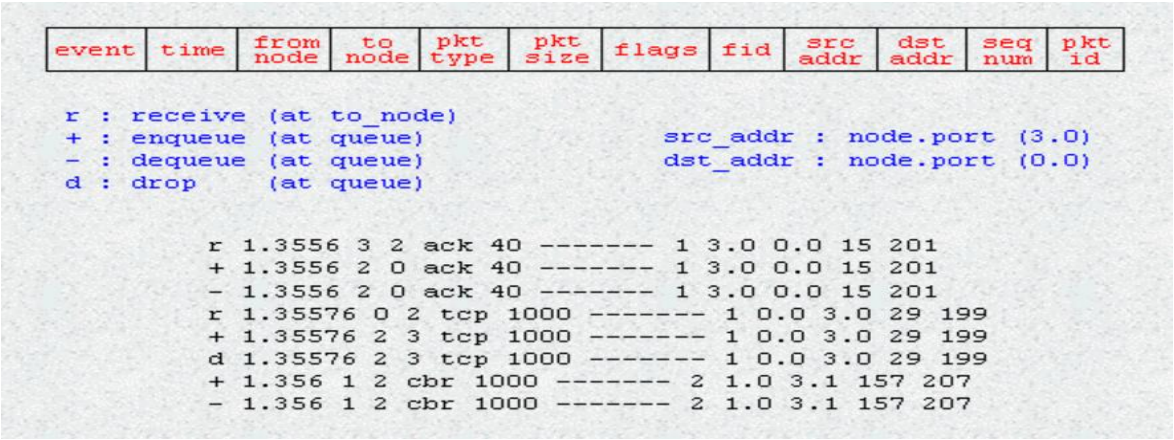


Figure 3.2 Trace format example [56]

3.3.1.3.3 NAM

NAM (Network AniMator) was one of the first tools to provide a general purpose, packet-level, network animation. With some preprocessing, NAM can also be used to visualize data taken directly from real network traces. NAM parses a trace file containing time-indexed network events to animate network traffic. Typically this trace is generated from an NS simulation, but it can also be generated by preprocessing from a live network. The NAM trace file contains all information needed for the animation: both the static network layout and dynamic events such as packet arrival, departures, and drops and link failure [57].

3.3.1.4 NS-MIRACLE

NS-MIRACLE (NS-Multi-InteRfAce Cross-Layer Extension) is a set of libraries designed to enhance the functionalities provided by the Network Simulator ns2. It provides an efficient and embedded engine for handling cross-layer messages and, at the same time, enables the coexistence of multiple modules within each layer of the protocol stack. For instance, multiple IP, link layers, MACs or physical layers can be specified and used within the same node. The NS-MIRACLE framework facilitates the implementation and the simulation of modern communication systems in ns2; moreover, due to its modularity, the code will be portable, reusable and extensible [58]. *Figure 3.3* illustrates the main features added [59] and that are outlined as follows:

- *Module*: a class designed to contain any protocols or modules that we usually place in stack in order to follow the OSI model
- *SAP* (Service Access Point): a channel to enable communication between a module of adjacent layers such as defined in OSI structure. It is also used to implement the tracking of simulation activity
- *ChSAP* (Channel SAP): an ad-hoc module for the channel which is connected to all the nodes.
- *CISAP* (Cross-Layer SAP): in charge of the delivery of the cross-layer messages from modules to the Node-Core
- *Node-Core*: is the structure in charge to dispatch all the CIMessage
- *Plugin*: is the father class of Module and it is designed to attach any module direct to the Node-Core external from the stack of modules. In Plugin are defined only the cross layers communication functionalities of a Module, so it is equipped with the ClSAP in order to be connected to the cross layer bus, placed in Node-Core.

This allows to Plugin to contain all that features that is difficult to put in a fixed position of the stack. This class provides the programmer the possibility to place external intelligence that has to collaborate with several modules in the stack and so manage cross-layer functionalities.

- *Bin* is a structure implemented for enhanced tracking. It is in charge to trace all the drops occurrences during the simulation. It is a child class of both SAP and ClSAP, so it inherits all the functions to trace both packets and cross layer messages dropping. Thanks to Bin, a programmer can understand also specific failure situations that normally are not covered by the trace of the sending messages.

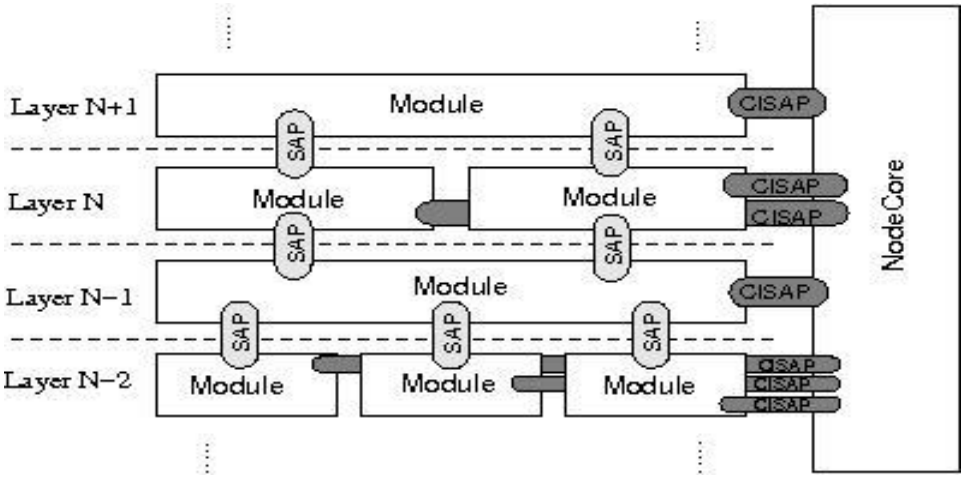


Figure 3.3 Example of a general multi-layer architecture within the MIRACLE framework [59]

3.3.1.5 DESERT Underwater

DESERT (DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols) is a set of C/C++ libraries to support the design and implementation of underwater network protocols. DESERT aims to extend NS-MIRACLE to provide several protocol stacks for underwater networks as well as the support routines required for the development of new protocols [11]. The flexibility and reliability to support the design and implementation of underwater network protocols by the DESERT Underwater libraries have been proven in [60]. A brief description of the built-in modules is also provided by authors as follows:

- Application layer: uwcbr and uwvbr
- Transport layer: uwudp and uwtp
- Network layer: uwstaticrouting, uwsun, uwicrp and uwip

- Data Link layer: uwmll, uwaloha, uw-csma-aloha, uwdacap, uwpolling, uw-t-lohi and uwsr
- Physical layer : uwmphypatch, uwmphy_modem, mfsk_whoi_mm, mpsk_whoi_mm, mgoby_whoi_mm and MS2C_EvoLogics
- Additional modules to simulate mobility:uwdriftposition, uwgmposition, wossgmmob3D, wossgroupmob3D

3.3.1.6 World Ocean Simulation system

World Ocean Simulation system (WOSS) was originally released in 2009 under a BSD 3-clause license, followed by two other releases that introduced several new features. It is a suite build upon and extends the capabilities of ns2 and NS-MIRACLE with the aim to improve underwater network simulations through a more realistic account of acoustic propagation. WOSS also provides a set of routines to facilitate many standard operations, like mobility management, the conversion among different coordinate systems (e.g, Cartesian to spherical), and the maintenance of the data structures required for the simulation of acoustic propagation [61]. Its large database consists of [62]: GEBCO bathymetric NetCDF Database, World Ocean Atlas 2009 SSP Database, DECK41 Sediment NetCDF Database, DECK41 NetCDF database, DECK41 marsden square NetCDF database, DECK41marsden coordinates NetCDF database, DECK41 Sediment selection logic, DECK41 geoacoustic parameters. The main advantage of the effort put into gathering this information into a WOSS-database is the simulation user only has to create an OTCL object where the latitude longitude and network size are set and WOSS handles the rest [11].

3.3.1.7 AQUASIM

Aquasim is a simulator for underwater sensor networks which is built on top of NS2 [63]. The effectiveness of Aqua-sim to simulate acoustic signal attenuation, packet collisions in underwater sensor networks, support three-dimensional deployment is discussed in [64]. The advantage with Aquasim is that it is a simulation package which runs in parallel with the CMU wireless package while relying on NS2 so as to be independent of the wireless package. *Figure 3.4* displays the relationship between Aquasim, CMU wireless package, and NS2. Currently organized into four folders (*uw common*, *uw mac*, *uw routing* and *uwtcl)*, Aqua-Sim follows the object-oriented design style of NS-2, and all network entities are implemented as classes in C++ [64][65][66].

Figure 3.5 shows a design of Aquasim architecture. The implementation of Aqua-Sim is briefly provided in [64] and it includes:

- Physical layer models:underwaterChannel, underwaterPhy
- MAC layer classes: Broadcast MAC, Aloha, *Tu*-MAC, and R-MAC. All these MAC protocols are derived from the same abstract base class "UnderwaterMac"
- Routing layer classes: Vector based routing protocol (VBF), Depth-base Routing (DBR) and Q-learning-based Routing (QELAR)
- Other classes: UnderwaterNode, GOD, UnderwaterSink

Among all those simulation tools, Aquasim is the one that has been chosen to go further in our work.

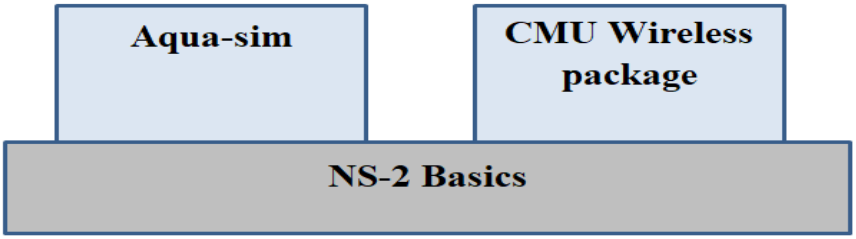


Figure 3.4 Relationship between Aqua-sim, CMU wireless package and NS-2 [64]

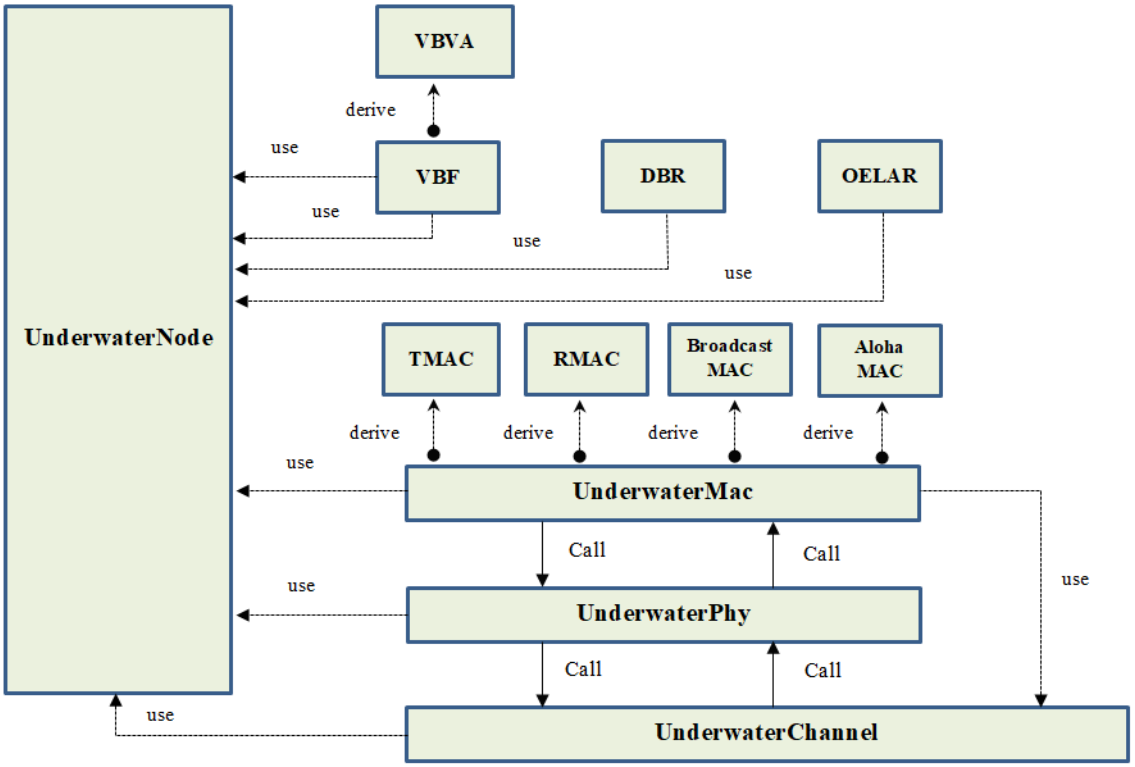


Figure 3.5 Class diagram of Aqua-sim [64]

3.3.2 Simulation setting

3.3.2.1 General design

The general design of the simulation set up is shown in *figure 3.6*. A cluster of 100 nodes is placed at the bottom of water with a dimension of 2000 m * 2000 m. A sink (Node 100) is placed roughly at the center of the cluster to collect data packets from sensor nodes. The distance range between the sink and the neighbor nodes is 84 m. The transmission range between sensor nodes varies between 75 and 84 m. Each sensor node was given an initial energy of 10k J. To route packets from the sensor nodes to the sink, we implemented the DSDV and AODV routing protocols. We then implemented BroadcastMac, UnderwaterPhy respectively for MAC layer protocol to access channel and physical layer. We set three (3) different sizes of data packet length of 2500, 1500 and 1000 bytes. Simulations were run for 300 simulator seconds. We also defined three (3) scenarios of 12, 25 and 50 nodes that transmit data simultaneously. To evaluate the performance of TCP, we will analyze the following metrics: the number of packet delivery and the retransmission ratio of segments that have been received successfully.

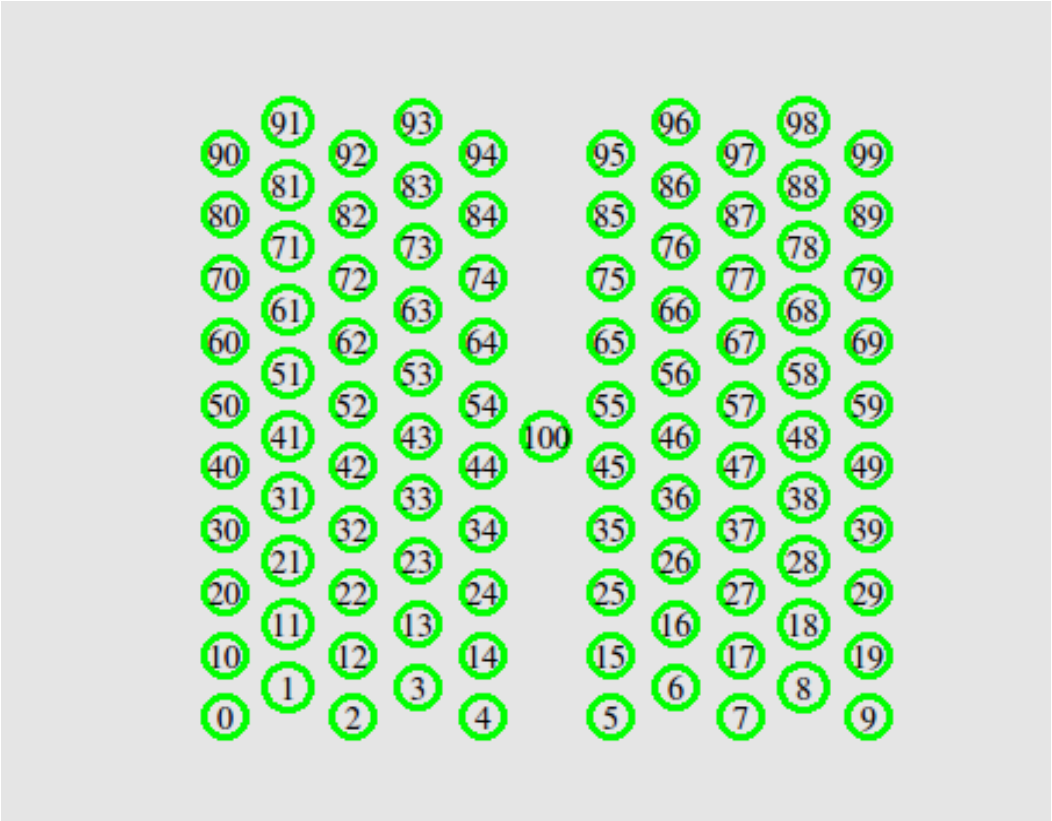


Figure 3.6 Overall design of the network

3.3.2.2 Description of simulation parameters

Table 3.1 shows the essential parameters of the simulation that have been set.

Table 3.1 Essential simulation parameters

Parameter	Value
Channel	UnderwaterChannel
Propagation	UnderwaterPropagation
PHY	UnderwaterPhy
Antenna	OmniAntenna
Distance	75m, 84m
Frequency	25kHz
MAC protocol	BroadcastMac
Mac_bit_rate	10kbps
Mac_packetheader_size	0
Mindelay	50 µs
Delay	25 µs
Routing protocols	DSDV
TCP agent sender	Newreno, U-Newreno
TCP agent receiver	TCPSink
Constant bit rate (CBR)	64 kbps

3.3.2.3 Description of TCP simulation parameters

Table 3.2 Simulation parameters of TCP

Parameter	Value
packetSize_	2500, 1500 and 1000 bytes
windowInit_	2 (for 2500 bytes), 3 (for 1500 bytes), 4 (for 1000 bytes)
window_	from 1 to 20 with incrementation by 1
rtxcur_init_	from 1 to 20 with incrementation by 1

The parameters contained in the above table represent the properties of TCP on which the variations of values will be made. This table can be divided into 2 types of options:

- Recommended options:

  These are the "packetSize_" and "windowInit_" properties referring respectively to SMSS and Initial Window (IW) in [40]. The choice of these parameters values are in line with the recommendations argued by the authors and are defined as follows:

  - packetSize_ = 2500 bytes corresponds to SMSS> 2190 bytes and imply IW = 2 * SMSS,

- packetSize_ = 1500 bytes corresponds to 1095 bytes <SMSS <= 2190 bytes and imply IW = 3 * SMSS
- packetSize_ = 1000 bytes corresponds to SMSS <= 1095 bytes and imply IW = 4 * SMSS

• Intuitive options

We undertook a cautious attitude. For this reason, the changes of the values of the parameters "window_" and "rtxcur_init" will be done with an increment step of 1 starting from 1 to 20. This incrementing value is intended to record the integer values for which the improvements of TCP are outstanding.

Table 3.3 Simplified simulation results

	Scenario	packetSize (bytes)	window_	rtxcur_init (sec)	Number of packets delivery with Newreno D.P	Number of packets delivery after both improvements	rtx_ratio with Newreno D.P	rtx_ratio after both improvements
DSDV	A	2500	2	8	7	10	3.3	3.1
		1500	16	7	7	7	1.9	1.6
		1000	16	8	8	10	3.4	2.9
	B	2500	1	1	0	27	0	3.7
		1500	5	1	16	23	4.4	3.9
		1000	1	15	16	20	4.1	4
	C	2500	1	19	10	19	4.1	2.5
		1500	3	18	0	22	0	3.5
		1000	4	17	0	22	0	3.2

Scenario defines the number of nodes that transmit traffic to the sink. Since the probability of contention and interference is high in a wireless network in general, it might be relevant to lead investigation based upon traffic volume resulting of transmission nodes.

(A) Corresponds to 12 transmitting nodes

(B) Corresponds to 25 transmitting nodes

(C) Corresponds to 50 transmitting nodes

packetSize: Sender Maximum Segment Size is the size of the largest segment that the sender can transmit [40].

window_ stands for Congestion Window (cwnd) upper bound. It refers to a TCP state variable that limits the amount of data a TCP can send.

rtxcur_init_: variable used to compute the initial RTT.

Number of packets delivery with Newreno D.P refers to the number of sequences received by the sink from the sensor nodes with the default parameters of Newreno.

Number of packets delivery after both improvements: refers to the number of sequences received by the sink from the sensor nodes after the improvement processes.

rtx_ratio with Newreno D.P: it is an average of the number of times a given segment is received by the sink when using Newreno with default parameters.

rtx_ratio after both improvements: it is an average of the number of times a segment is received by the sink after improvement processes.

DP: Default parameters of 'window_' and 'rtxcur_init_'

3.3.3 Performance evaluation

3.3.3.1 Metrics

The determination of the metrics to use was inspired by the observation of the trace file obtained during some simulations carried out. This simulation was performed with the TCP default settings. In fact, we have observed two particularities:

- in some cases, we did not record packets delivery
- in others, we did record packets delivery with a high rate of duplication

So, this led us to think about two metrics:

1) *Packet delivery:* this metric responds to the first problem of the trace file listed above and therefore the need to show that the data transmission in the UWSN with TCP as a transport protocol is feasible. And second, to show that the reception of packets can grow if the necessary adjustments are well developed. It is good to remember that talking about a "packet" refers to a "sequence number" as we deal with TCP. It is, therefore, a question of counting the number of sequence number received during the simulation time. Thus, the greater the number of sequence number received, the better the TCP performance will be judged.
2) *Retransmission ratio*: it addresses the second problem of the trace file. It defines on average the number of times a sequence has been received during the simulation. Thus, the smaller this indicator is, the better the TCP performance is, because it shows that there is less retransmission and consequently an energy efficiency which represents one of the challenges of the UWSN

3.3.3.2 Results and analysis

Simulation results will be displayed. It will be shown all the results in scenario A, then follows by scenario B and finally scenario C. Subsequently an analysis is proposed relatively to the scenarios. The reason why we have performed many scenarios is that

we could not predict the behavior of TCP in a UWSN environment. Indeed, we could not know if there is a relationship between the number of source nodes that induce the number of packets transmitted and the number of delivered packets.

3.3.3.2.1 Results of scenario A

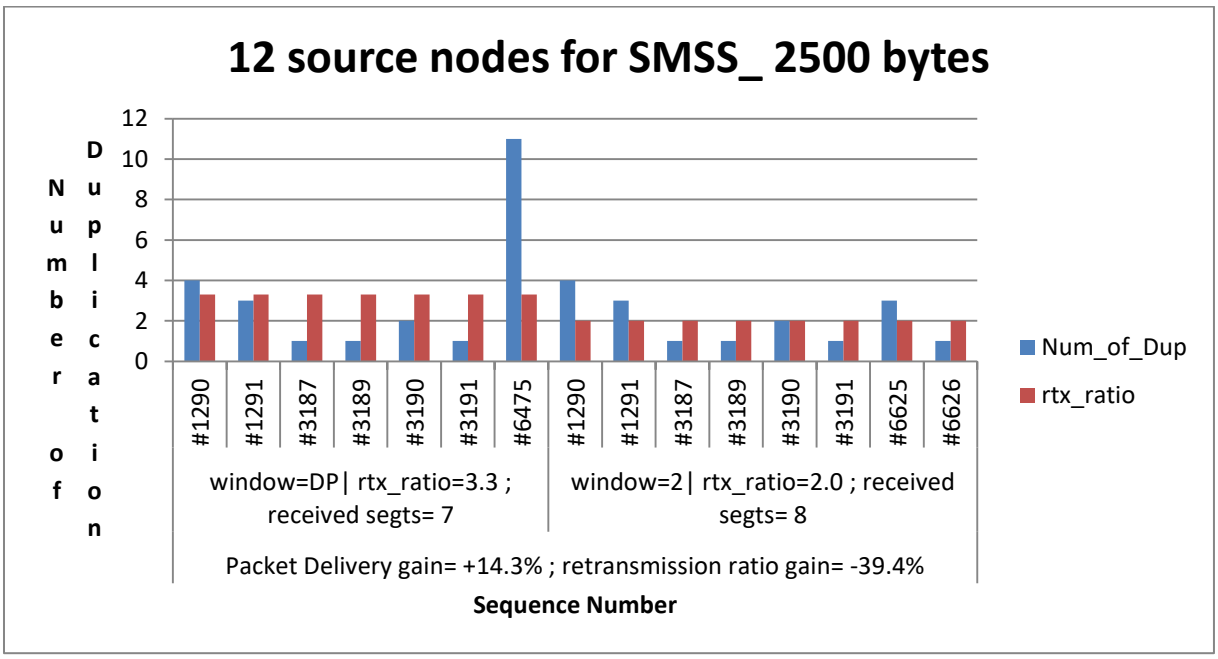


Figure 3.7 Impact of controlling maximum window in scenario A while SMSS_ 2500 bytes

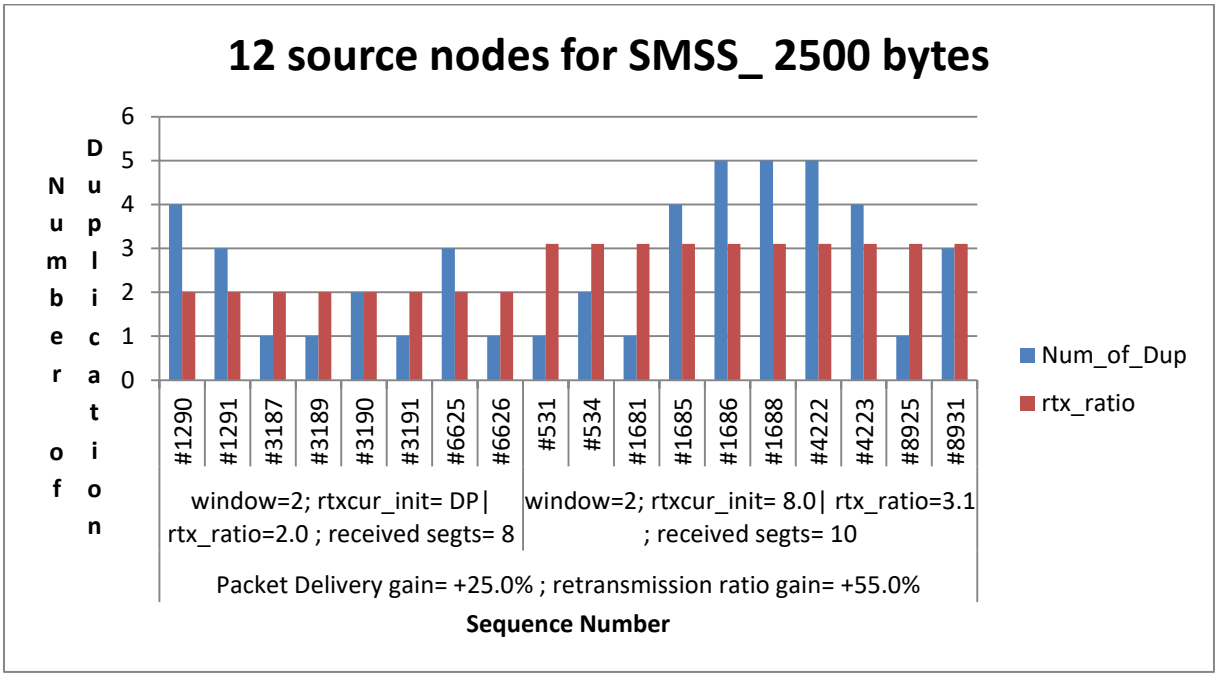


Figure 3.8 Impact of appropriate "rtxcur_init" value in scenario A while SMSS_ 2500 bytes

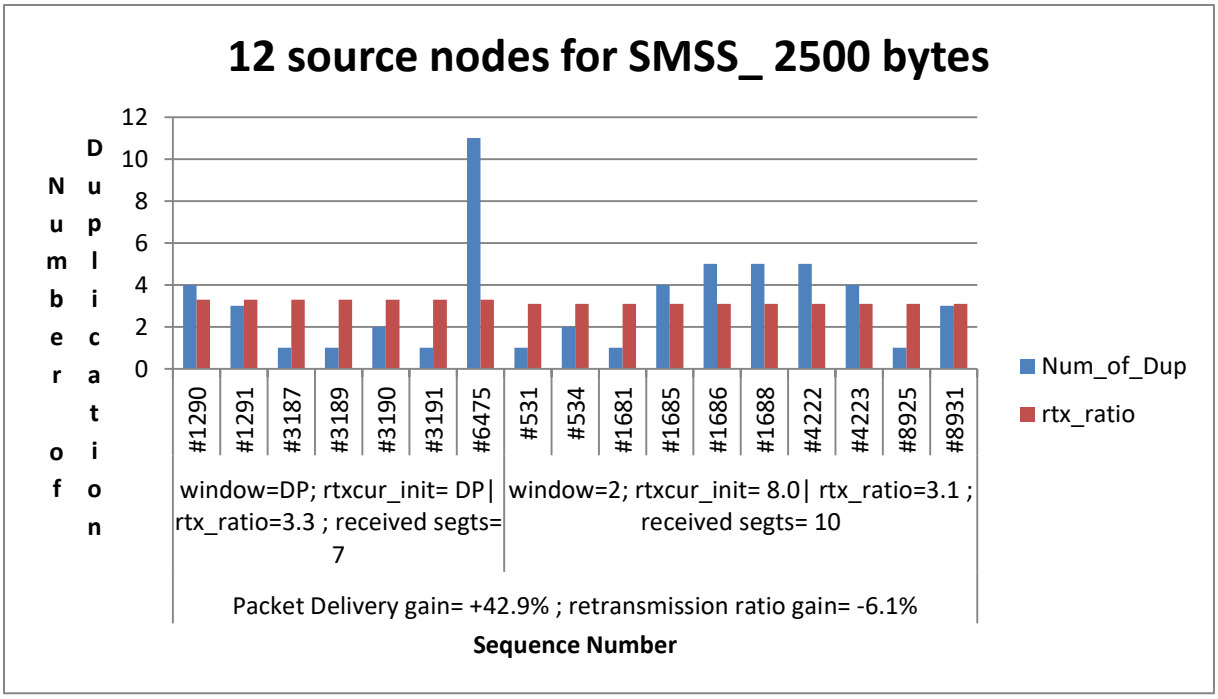


Figure 3.9 Illustration of TCP performance before and after both two combined improvements in scenario A while SMSS_ 2500 bytes

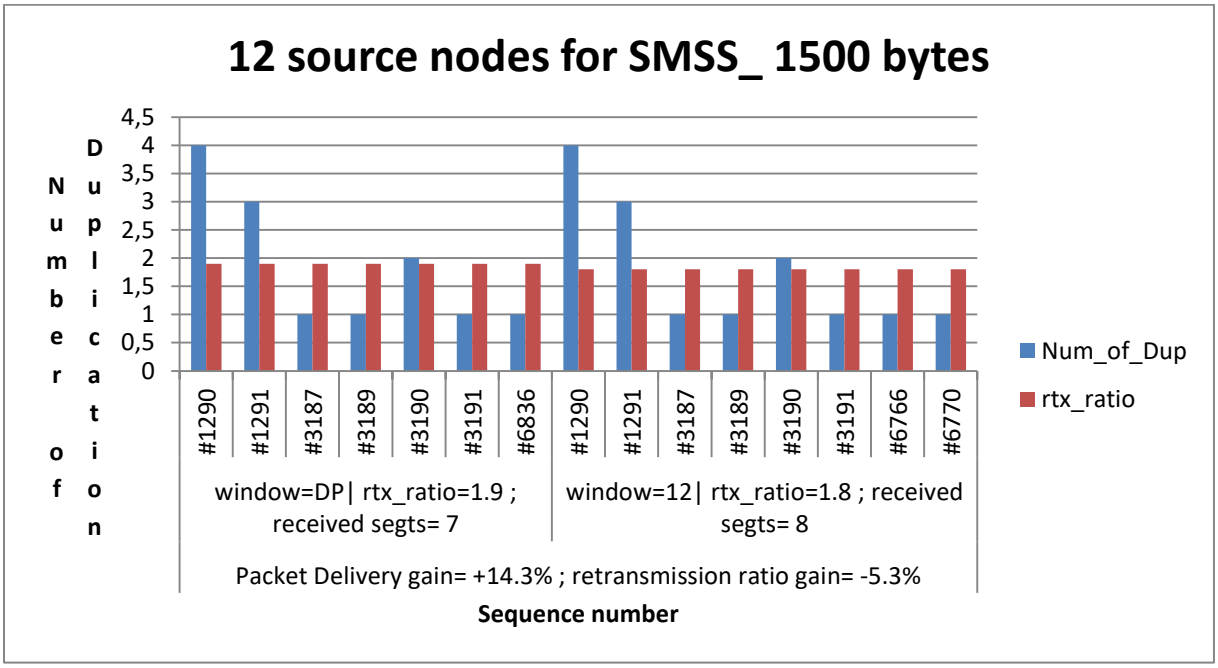


Figure 3.10 Impact of controlling maximum window in scenario A while SMSS_ 1500 bytes

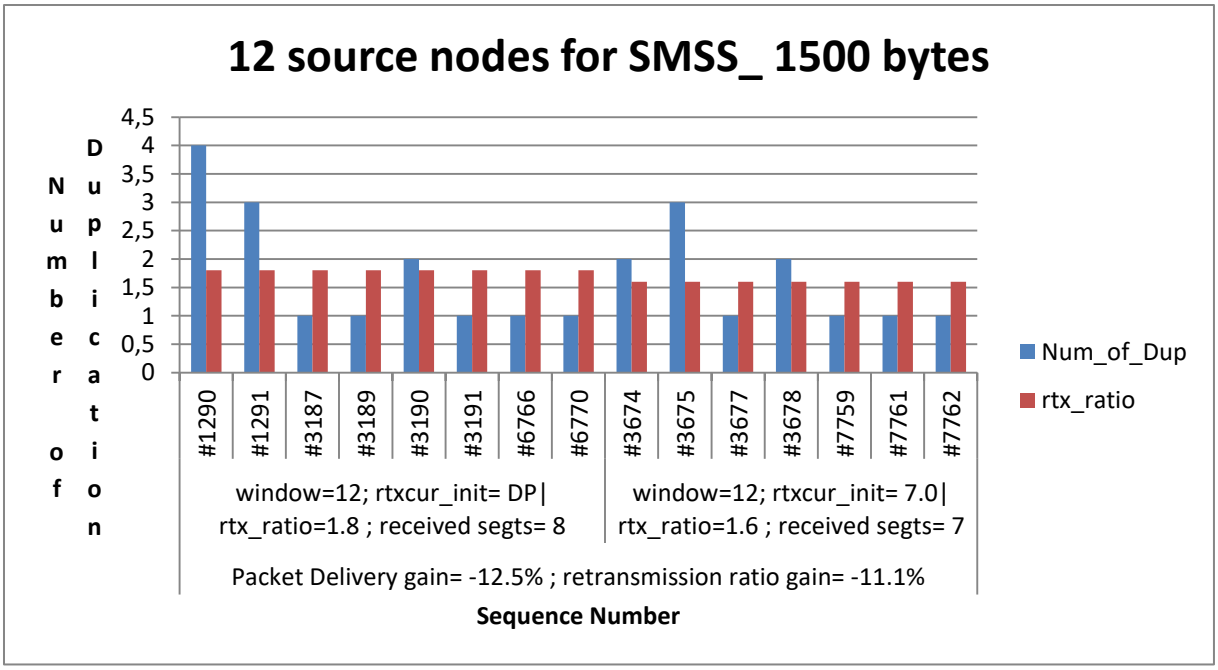


Figure 3.11 Impact of appropriate "rtxcur_init" value in scenario A while SMSS_ 1500 bytes

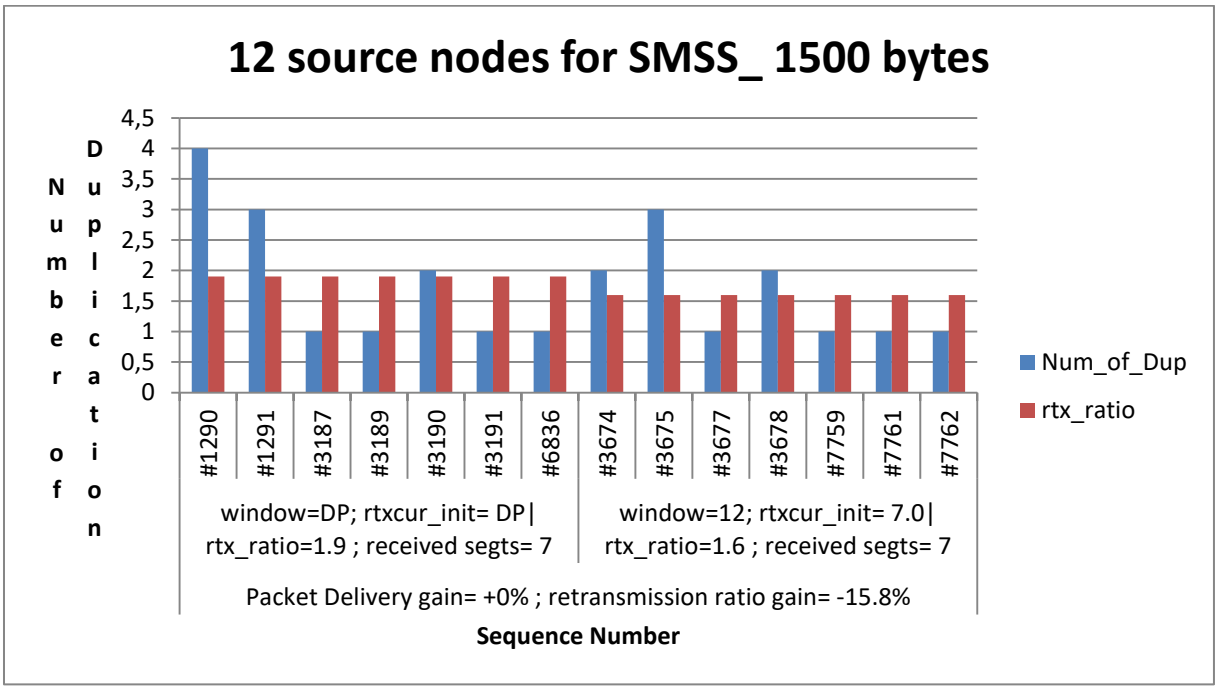


Figure 3.12 Illustration of TCP performance before and after both combined improvements in scenario A while SMSS_ 1500 bytes

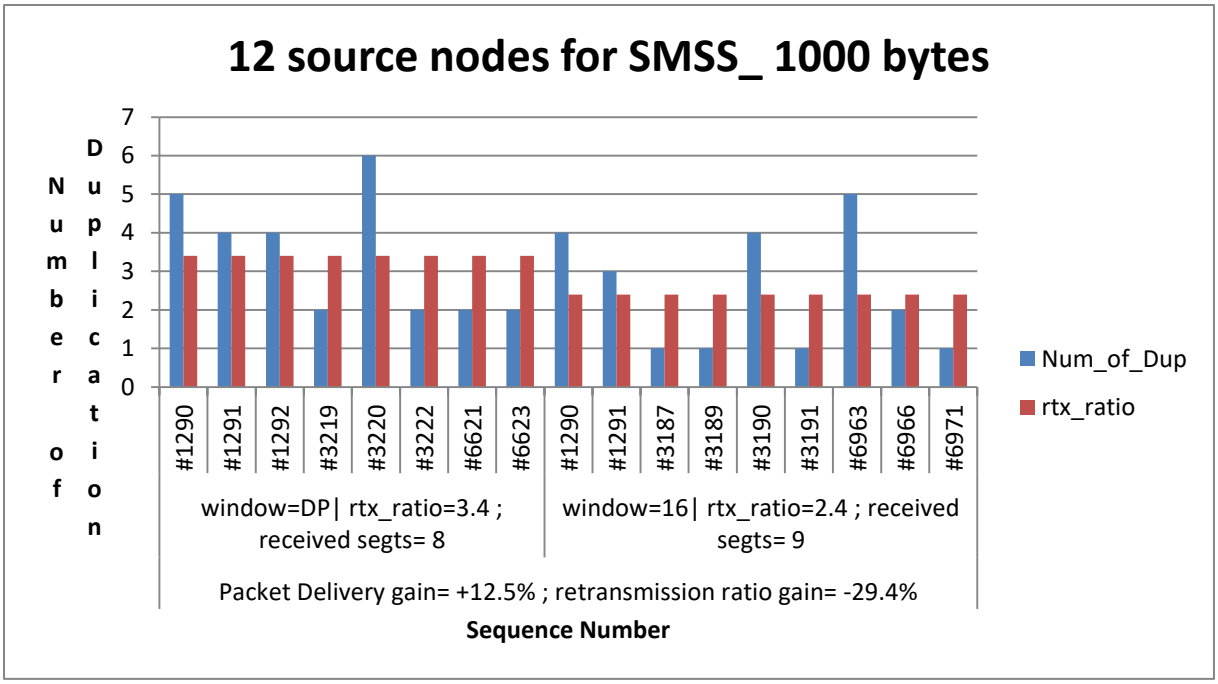


Figure 3.13 Impact of controlling maximum window in scenario A while SMSS_ 1000 bytes

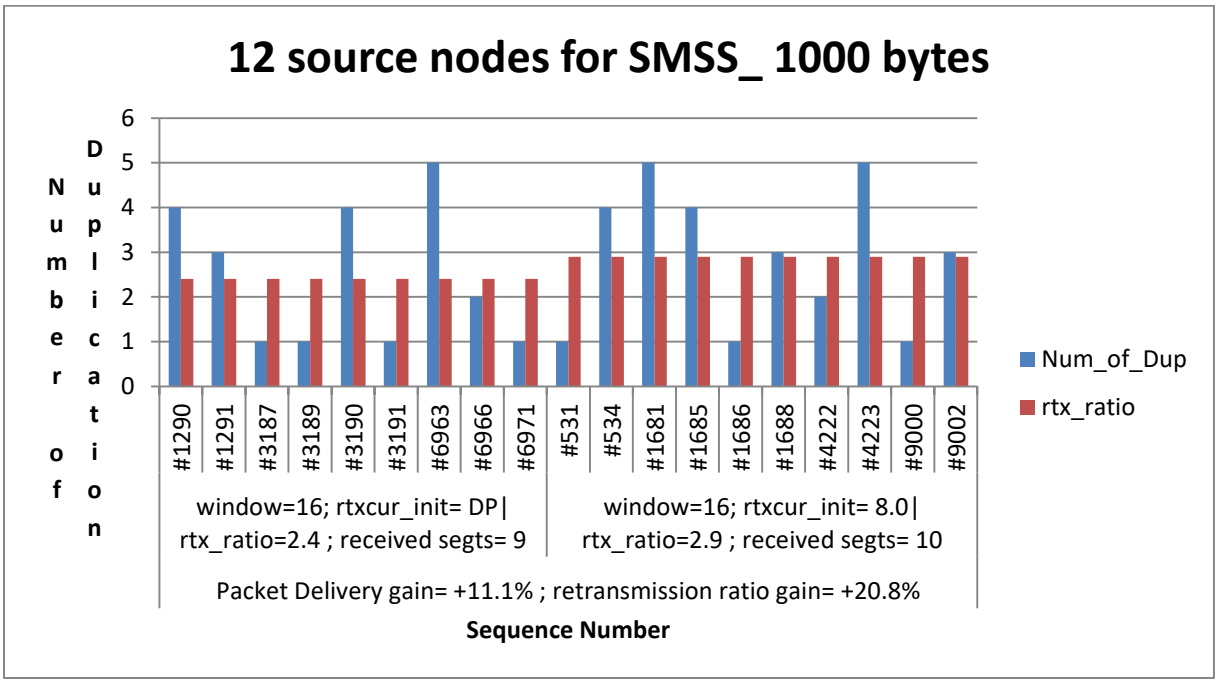


Figure 3.14 Impact of appropriate "rtxcur_init" value in scenario A while SMSS_ 1000 bytes

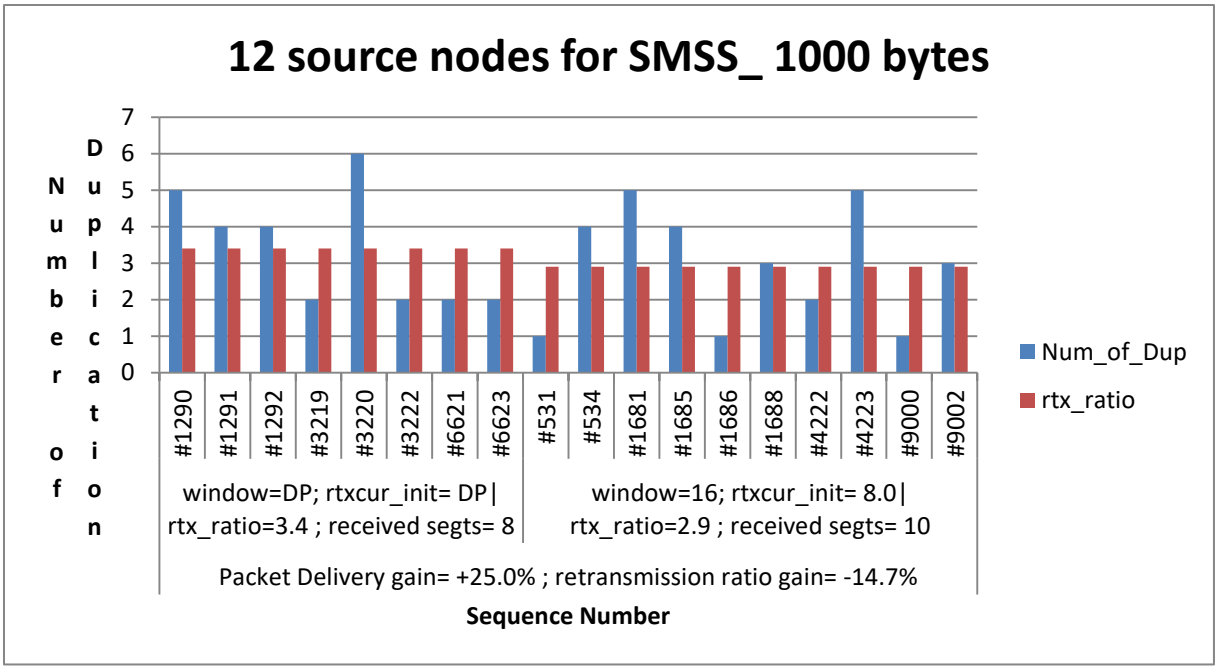


Figure 3.15 Illustration of TCP performances before and after the two combined improvements in scenario A while SMSS_ 1000 bytes

Packet delivery

Regarding the upper bound parameter of the sending window, all cases show a relative increase by +14.3%, +14.3%, +12.5% respectively illustrated in *figures 3.7, 3.10 and 3.13*.We can see that the 2500 bytes case has an optimal performance for a very small window size estimated at 2. The two remaining cases show optimal performance for larger values of the window size estimated at 12 and 16 respectively for 1500 and 1000 bytes cases. What would be the performance of TCP with a window set to 2 for the latter two cases? The answer is that we also get increases of +14.3% and +28.6%, which is acceptable but not sufficient when adding the retransmission ratio which gets higher. The relatively large size of the maximum size of the window for the two smallest SMSS values could be explained by the capacity of the channel to properly manage the network flow in terms of bandwidth and also the queues at the intermediate nodes.

The second improvement level is based on the first level of improvement which is the suitable upper limit of the sending window associated with the default "rtxcur_init_" parameter. Thus, after having found the appropriate value of the rtxcur_init for which the optimal performance of TCP is obtained at this stage, the comparison between the two levels of improvement is carried out. Improvements are also seen at this level as we can see for the 2500 and 1000 bytes cases respectively, there is an increase of +25% (*cf. figure*

3.8) and + 11.1% *(cf. figure 3.14)*.On the other hand, the 1500 bytes case shows a decrease of -12.5% (*cf. figure 3.11*), which may seem to be a disadvantage because the possibility of improving the number of delivered packets (up to 9) is possible for rtxcur_init values at { 2; 8} but the disadvantage is the retransmission rate which proves to be high. The suitable value of the 'rtxcur_init' approaching 7 or 8 sec, which is more than twice as much in the terrestrial environment, could be due to the characteristics of the channel which predicts a long and variable duration of the RTT.

Retransmission ratio

The ideal objective is to observe a decrease in the retransmission rate when the gain of received packets is positive. This is the case here with the upper limit of the sending window: -39.4% (*cf. figure 3.7*), -5.3% (*cf. figure 3.10*) and -29.4% (*cf. figure 3.13*) respectively for 2500, 1500 and 1000 bytes cases.

Moreover, taking into consideration the 'rtxcur_init' parameter presents divergent results. When the 1500 bytes case shows an improvement with a change from 1.8 to 1.6 or -11.1% (*cf. figure 3.11*), the other two cases show impairments of +55% and +20.8% respectively illustrated by *figures 3.8 and 3.14*. These situations show the difficulty of correctly estimating the value of 'rtxcur_init' in this configuration.

Finally, comparing the performance of TCP before and after the improvements, get us closer to the ideal sought (*cf. figure 3.9 and figure 3.15*) except for 1500 bytes case where the number of delivered packets remains unchanged but evolves in terms of retransmission ratio by -15.8% (*cf. figure 3.12*)

3.3.3.2.2 Results of scenario B

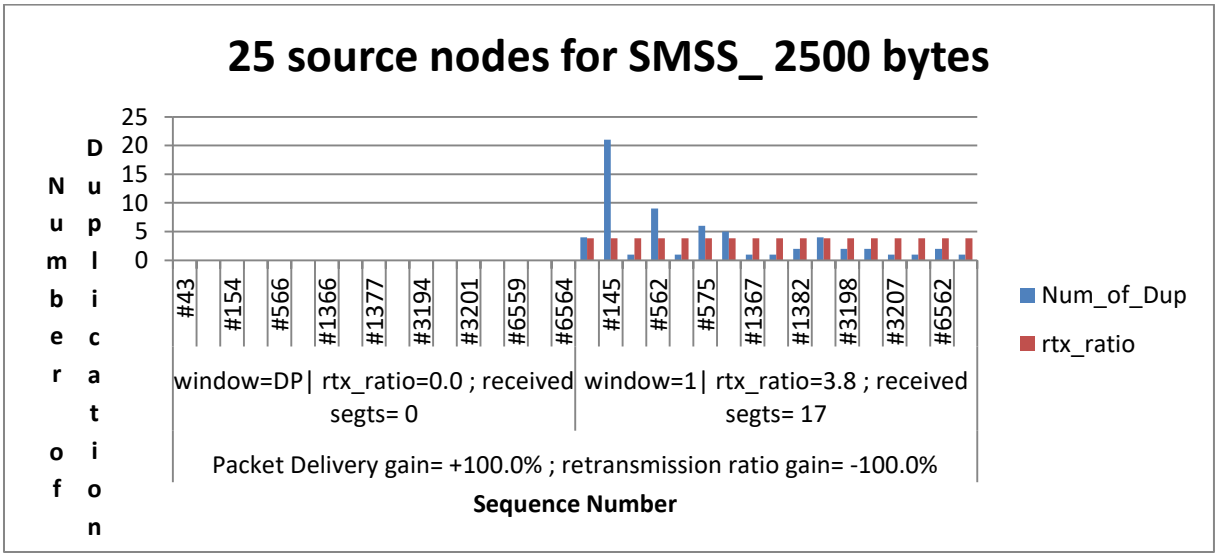


Figure 3.16 Impact of controlling maximum window in scenario B while SMSS_ 2500 bytes

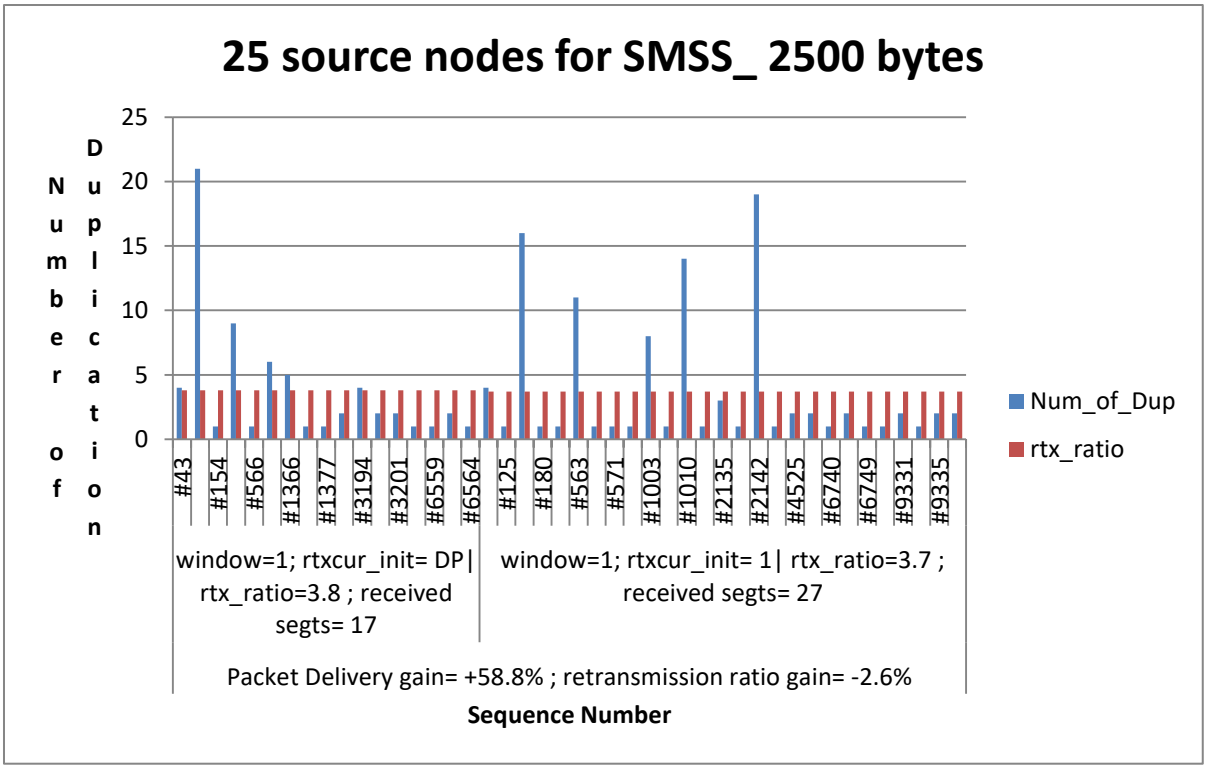


Figure 3.17 Impact of appropriate "rtxcur_init" value in scenario B while SMSS_ 2500 bytes

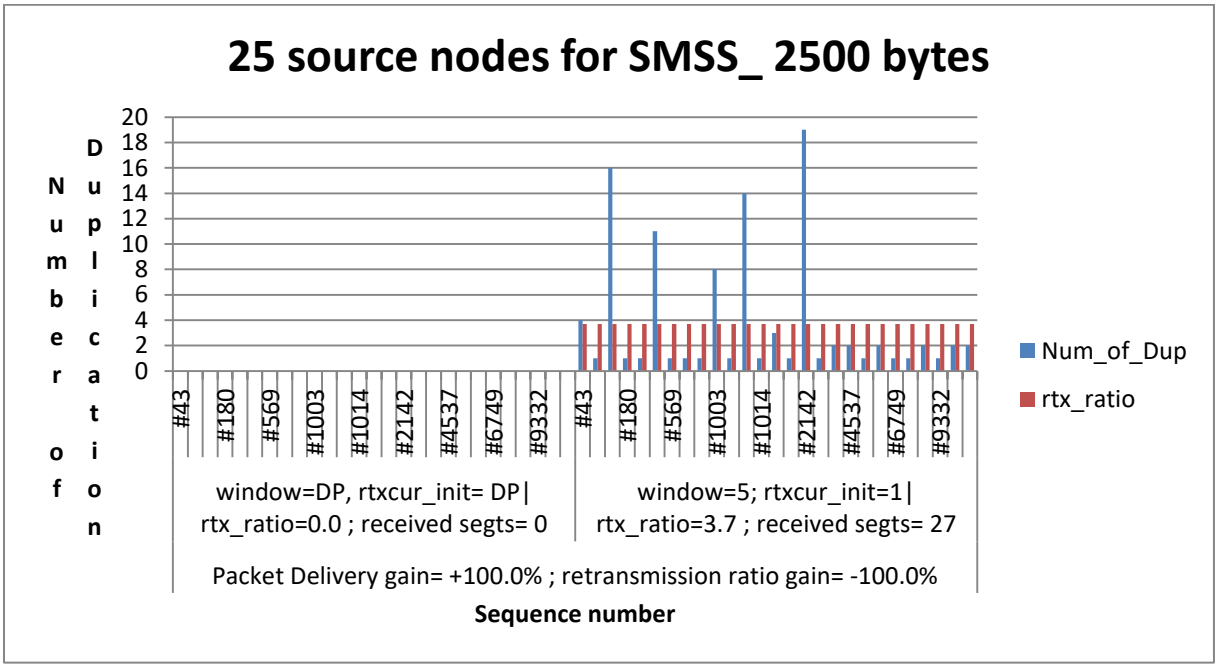


Figure 3.18 Illustration of TCP performance before and after both combined improvements in scenario B while SMSS_ 2500 bytes

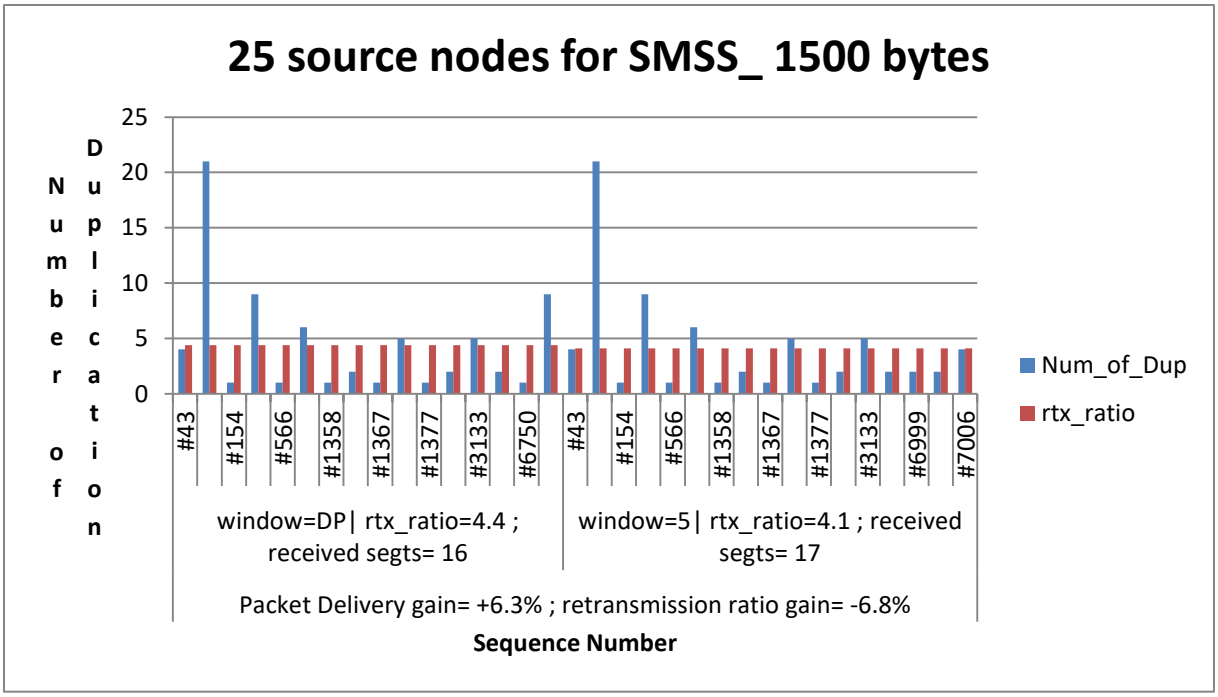


Figure 3.19 Impact of controlling maximum window in scenario B while SMSS_ 1500 bytes

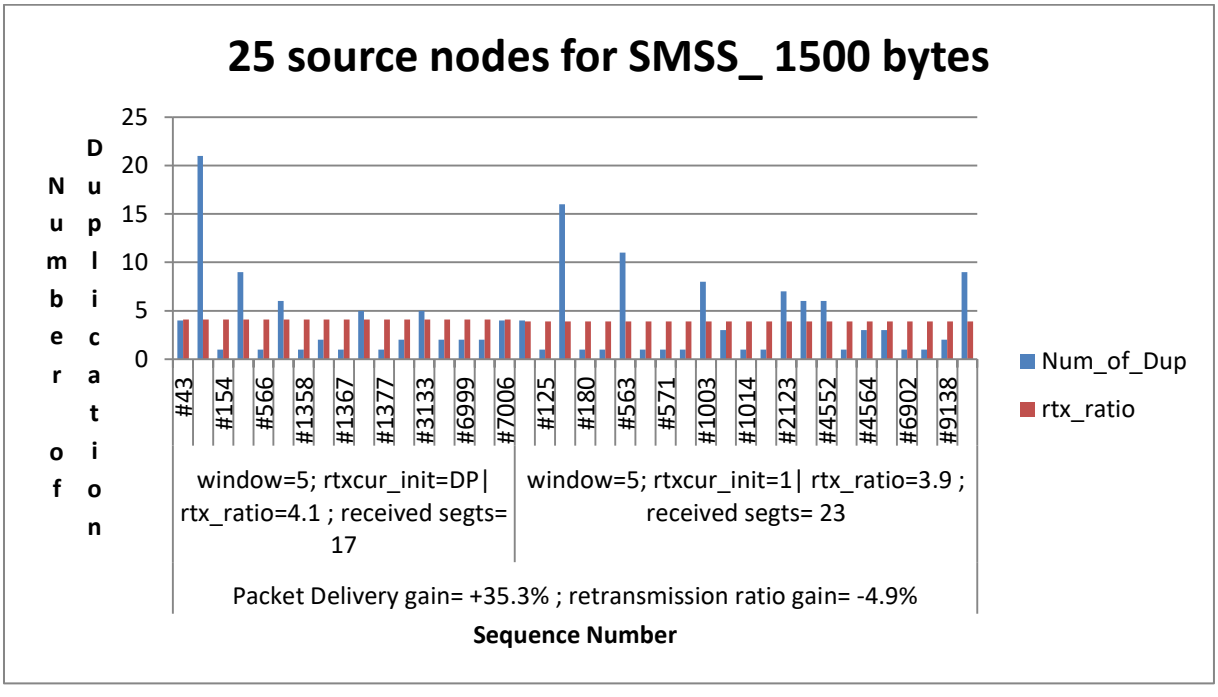


Figure 3.20 Impact of appropriate "rtxcur_init" value in scenario B while SMSS_ 1500 bytes

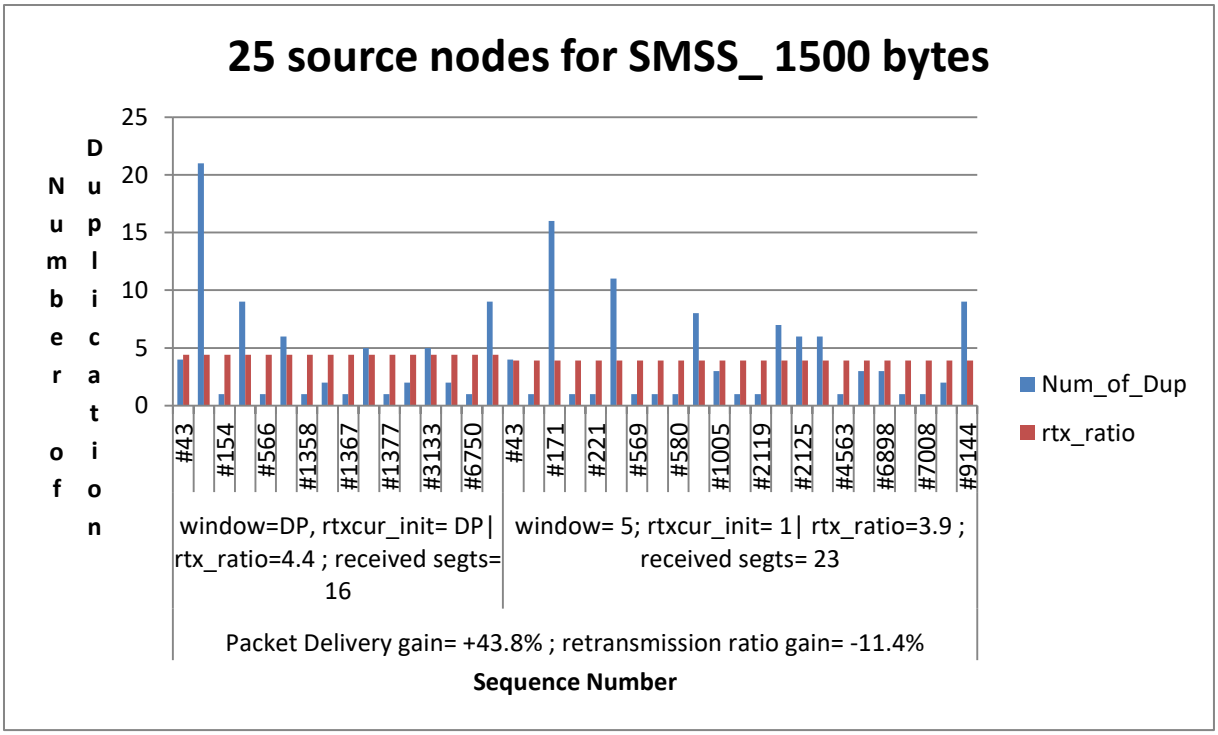


Figure 3.21 Illustration of TCP performance before and after both combined improvements in scenario B while SMSS_ 1500 bytes

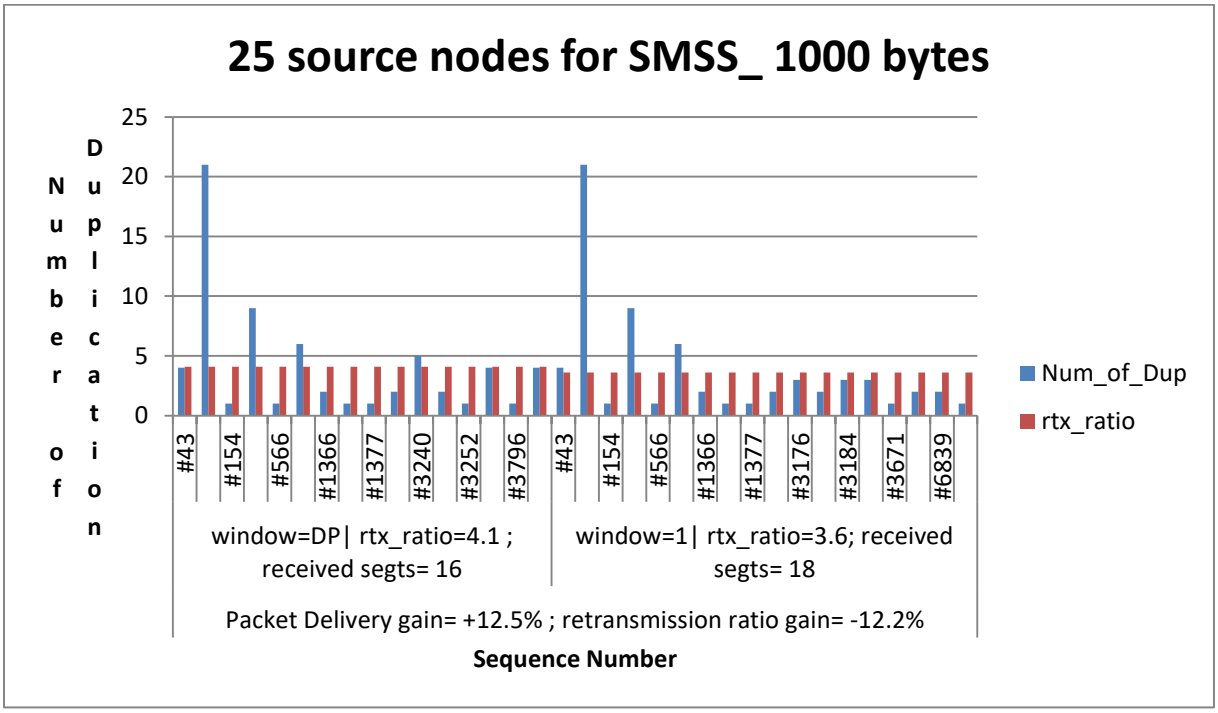


Figure 3.22 Impact of controlling maximum window in scenario B while SMSS_ 1000 bytes

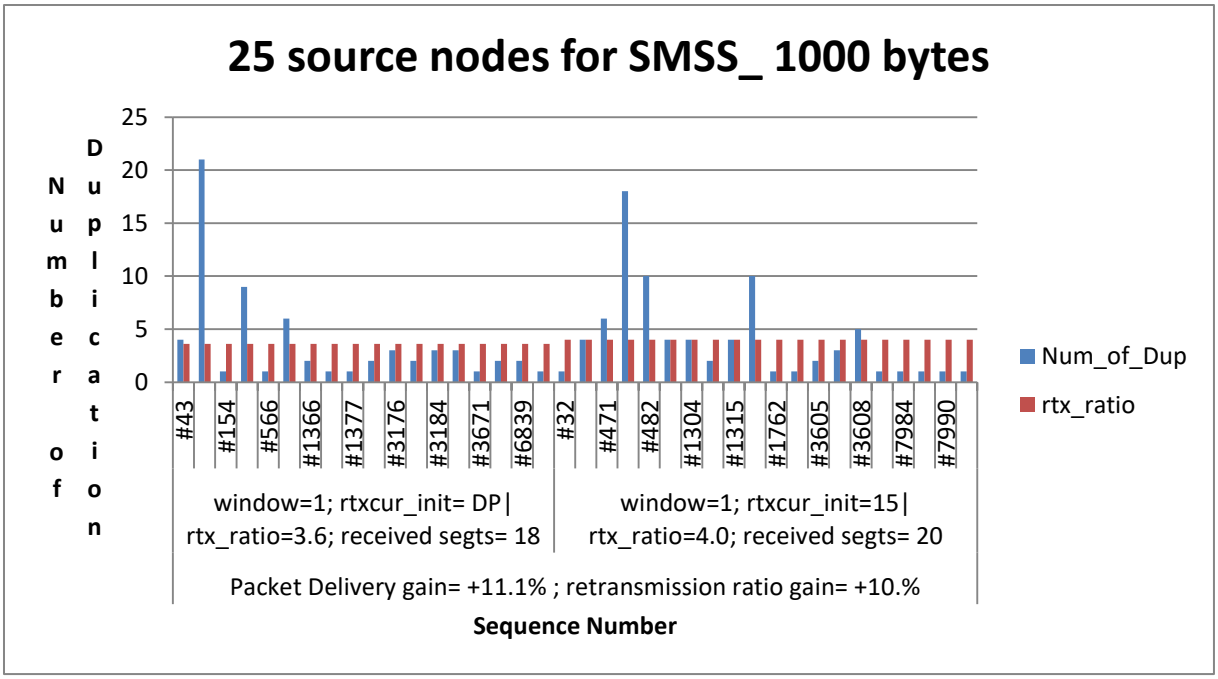


Figure 3.23 Impact of appropriate "rtxcur_init" value in scenario B while SMSS_ 1000 bytes

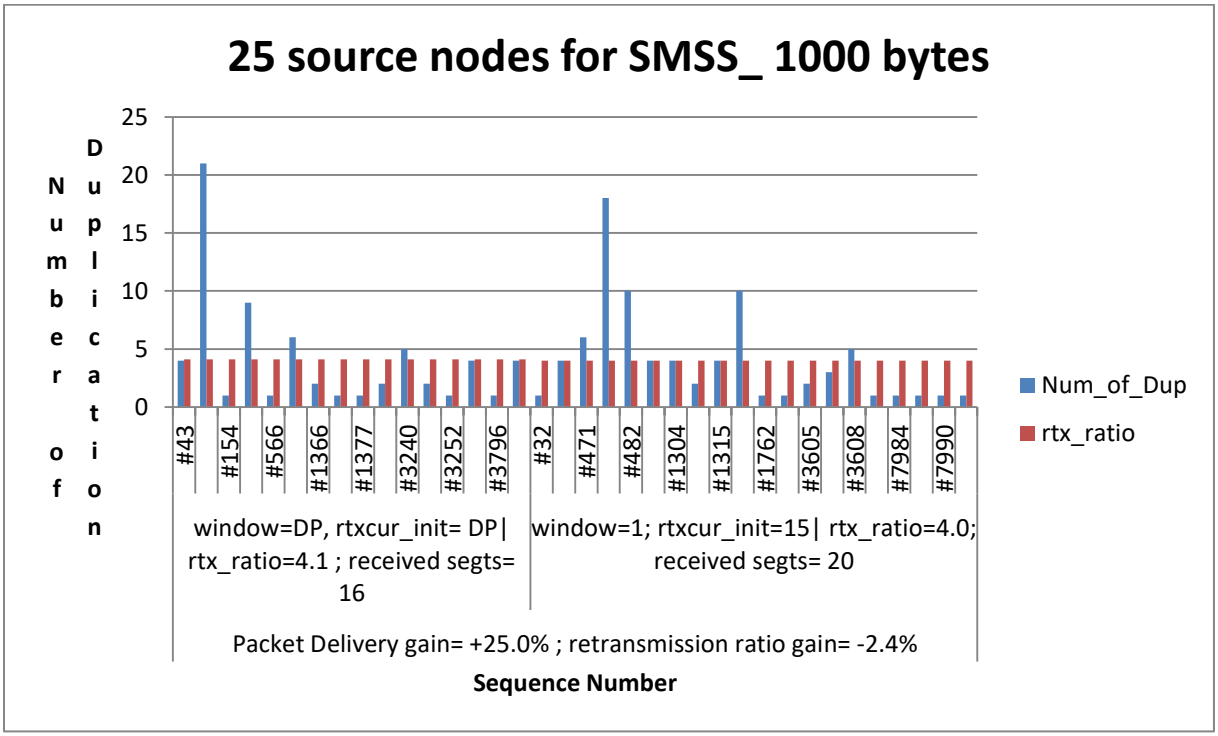


Figure 3.24 Illustration of TCP performance before and after both combined improvements in scenario B while SMSS_ 1000 bytes

Packet delivery

Figure 3.16 shows the relevance of controlling the congestion window limit. Indeed, the simulation with the TCP default parameters did not record any segment reception. Conversely, the reception of segments is obtained for a limit fixed to 1 for this case. The same applies to the other two cases with improvements of +6.3% (*cf. figure 3.19*) and +12.5% (*cf. figure 3.22*). It should be noted that the optimal improvements at this stage are achieved for small values of the window size upper bound between {1; 5}. An explanation could be that the traffic that is related to the source nodes increases, which would cause more risk of collision or interference hence the need to inject less traffic at the same time. In addition, the improvement of the packet delivery is observed for the appropriate value of rtxcur_init as well. We note evolution from:

- ✓ 17 to 27 that represents a gain of +58.8% (*cf. figure 3.17*)
- ✓ 17 to 23 that is a gain of +35.3% (*cf. figure 3.20*)
- ✓ 18 to 20 that is a gain of +11.1% (*cf. figure 3.23*)

The value of rtxcur_init is uniformly 1sec for the first two cases (*cf. figure 3.17* and *figure 3.20*), while for the 1000 bytes case this value is 15 sec. However, in the 1000 bytes case, the best performance with a small value of this parameter is reached at 2 sec with the number of received segments remaining at 18 but the reverse is the retransmission ratio which increases considerably.

Retransmission ratio

The principle of reducing the retransmission rate is achieved in all cases when it is applied:

- Control of the upper window limit as follows: -100% (*cf. figure 3.16*), -6.8% (cf. *figure 3.19*) and -12.2% (cf. *figure 3.22*)
- Adjustment of the value of rtxcur_init as follows: -2.6% (cf. *figure 3.17*) and -4.9% (cf. *figure 3.20*). However, an exceptional increasing by +10% is observed for the 1000 bytes case (cf. *figure 3.23*)

In the end, this scenario presents improvements in terms of packets delivery gain of all cases as follows:

- ✓ *Figure3.18* shows progress from 0 to 27 received packets
- ✓ *figure 3.21* shows enhancement from 16 to 23 received packets or +43.8%
- ✓ *figure 3.24* shows improvement from 12 to 20 or received segments or +25%

Similarly, there was an improvement in the retransmission ratio gain of -11.4% (cf. *figure 3.21*), -2.4% (cf. *figure 3.24*) respectively for the 1500 bytes and 1000 bytes cases. The

exceptional case of 2500 bytes does not allow us to evaluate the gain because before the improvement no recording had been made so as to determine the retransmission ratio.

3.3.3.2.3 Results of scenario C

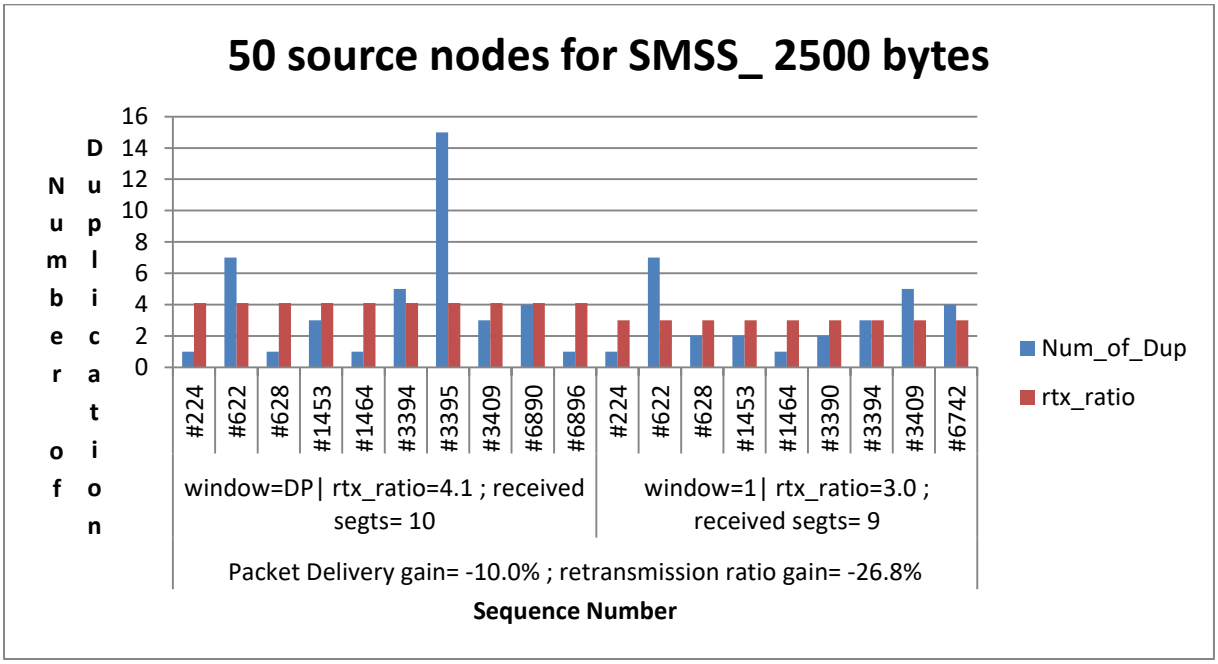


Figure 3.25 Impact of controlling maximum window in scenario C while SMSS_ 2500 bytes

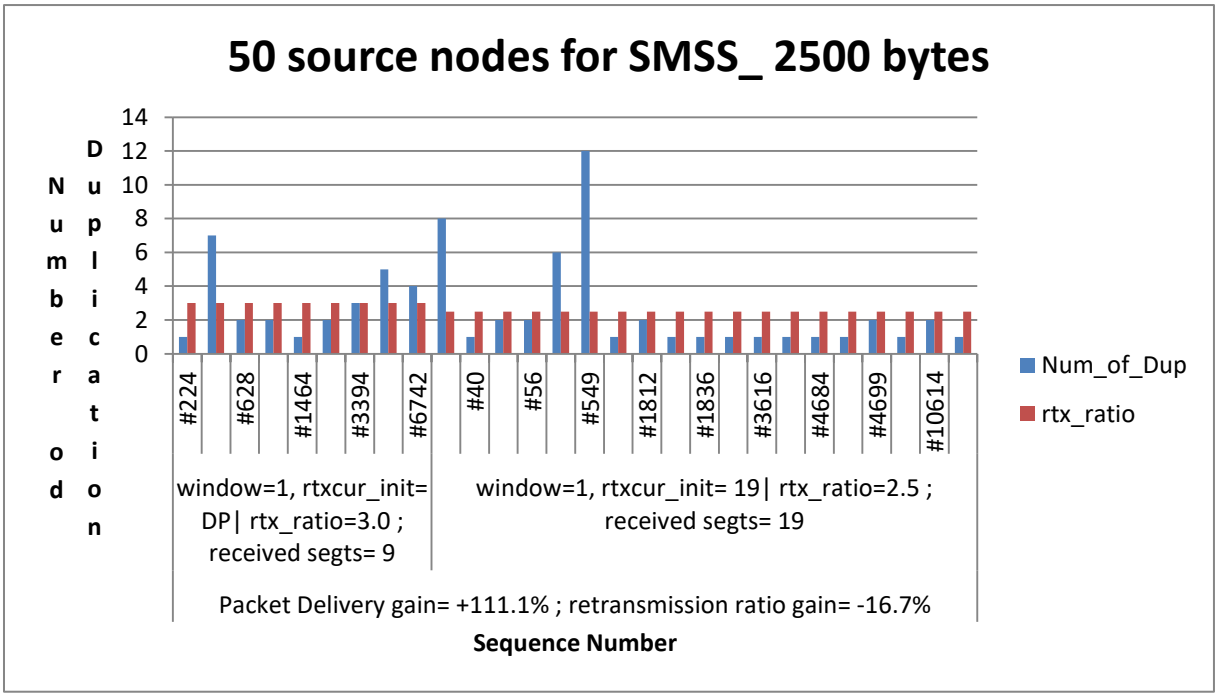


Figure 3.26 Impact of appropriate "rtxcur_init" value in scenario C while SMSS_ 2500 bytes

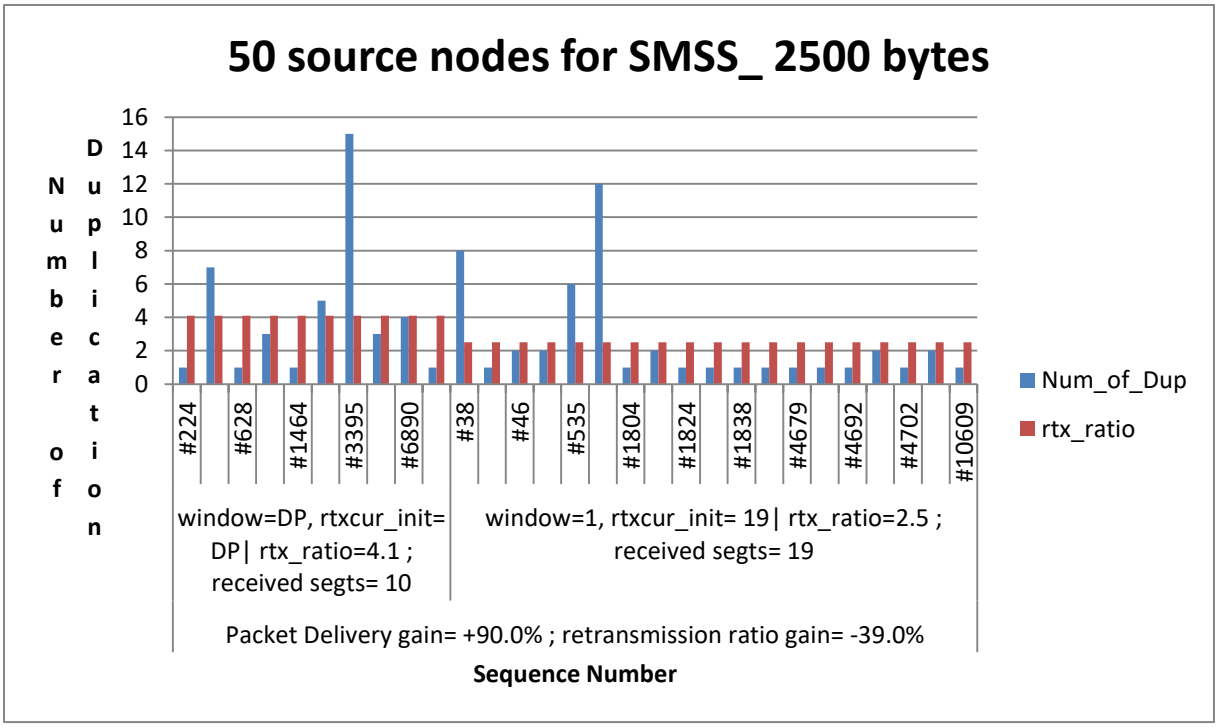


Figure 3.27 Illustration of TCP performances before and after both combined improvements in scenario C while SMSS_ 2500 bytes

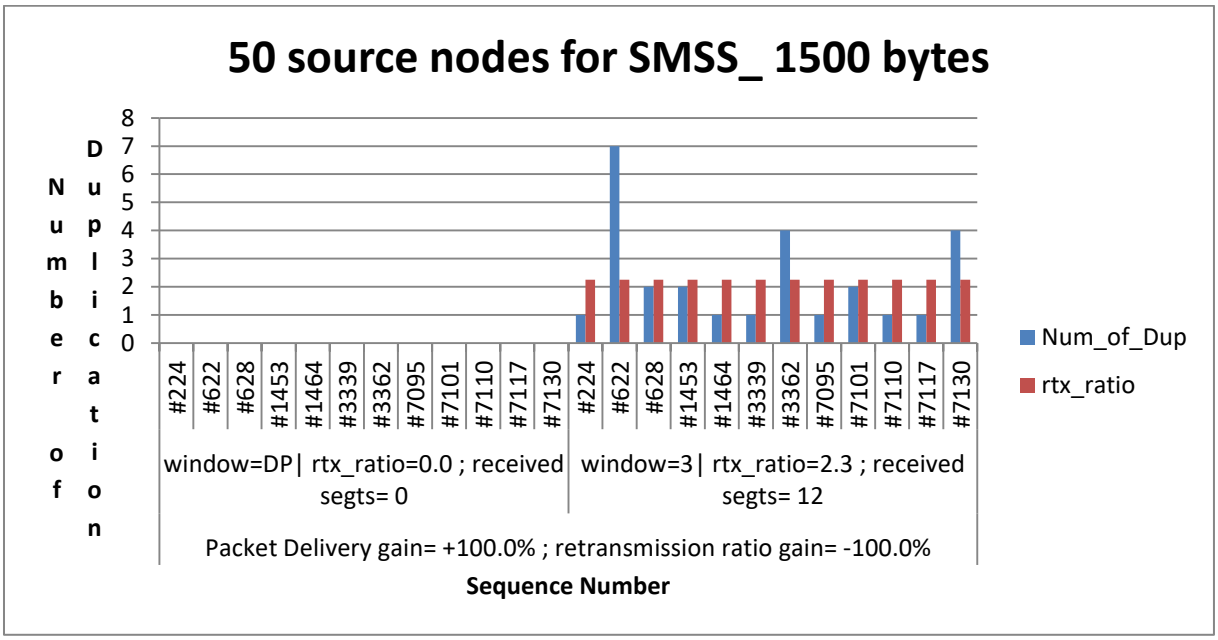


Figure 3.28 Impact of controlling maximum window in scenario C while SMSS_ 1500 bytes

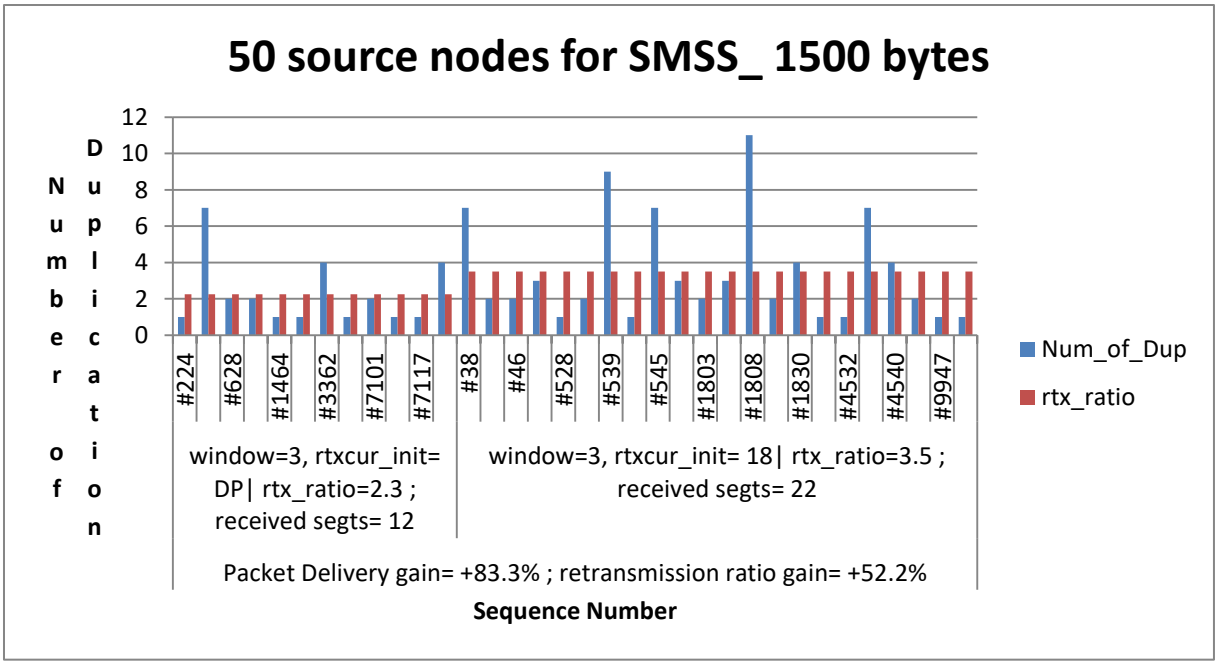


Figure 3.29 Impact of appropriate "rtxcur_init" value in scenario C while SMSS_ 1500 bytes

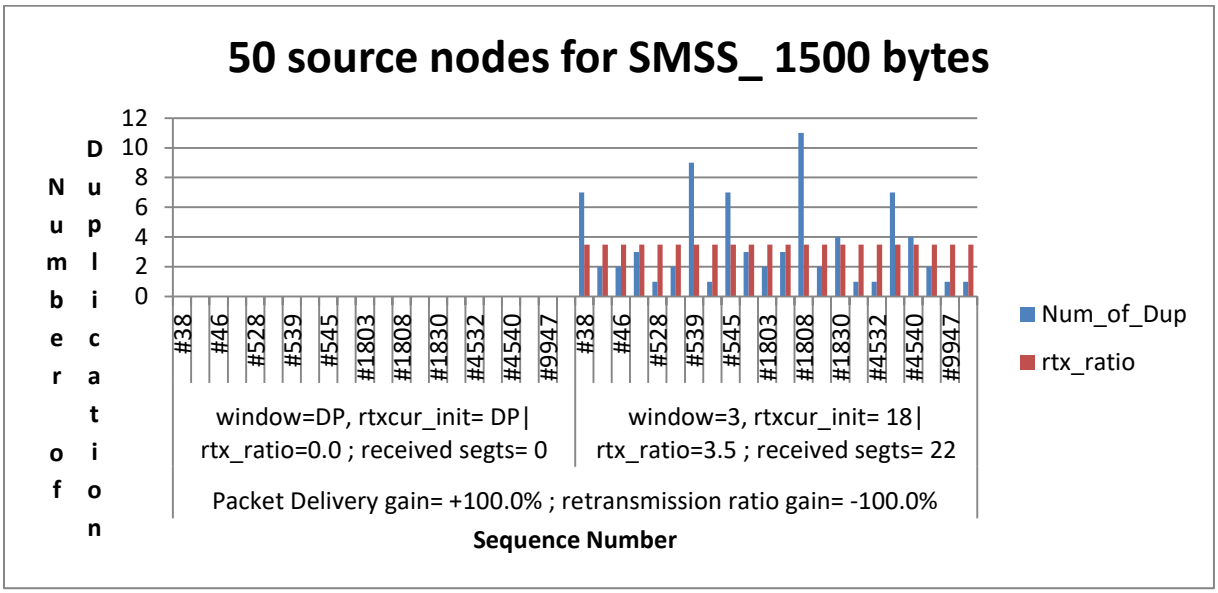


Figure 3.30 Illustration of TCP performances before and after both combined improvements in scenario C while SMSS_ 1500 bytes

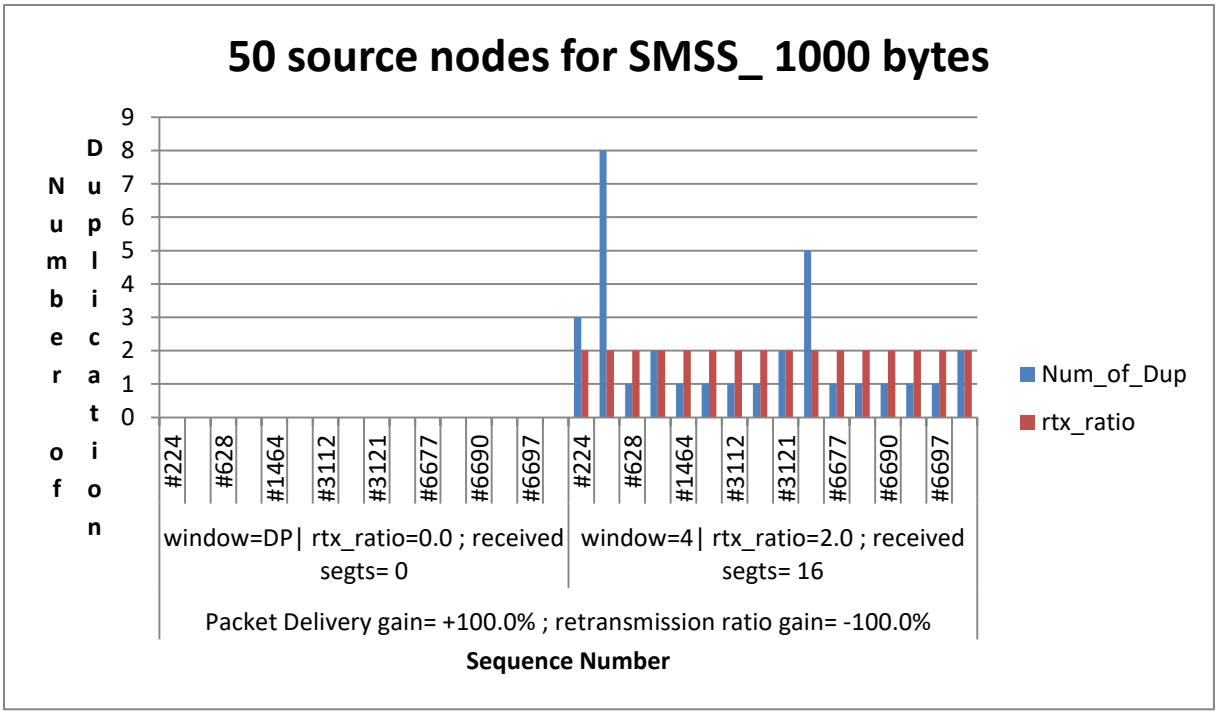


Figure 3.31 Impact of controlling maximum window in scenario C while SMSS_ 1000 bytes

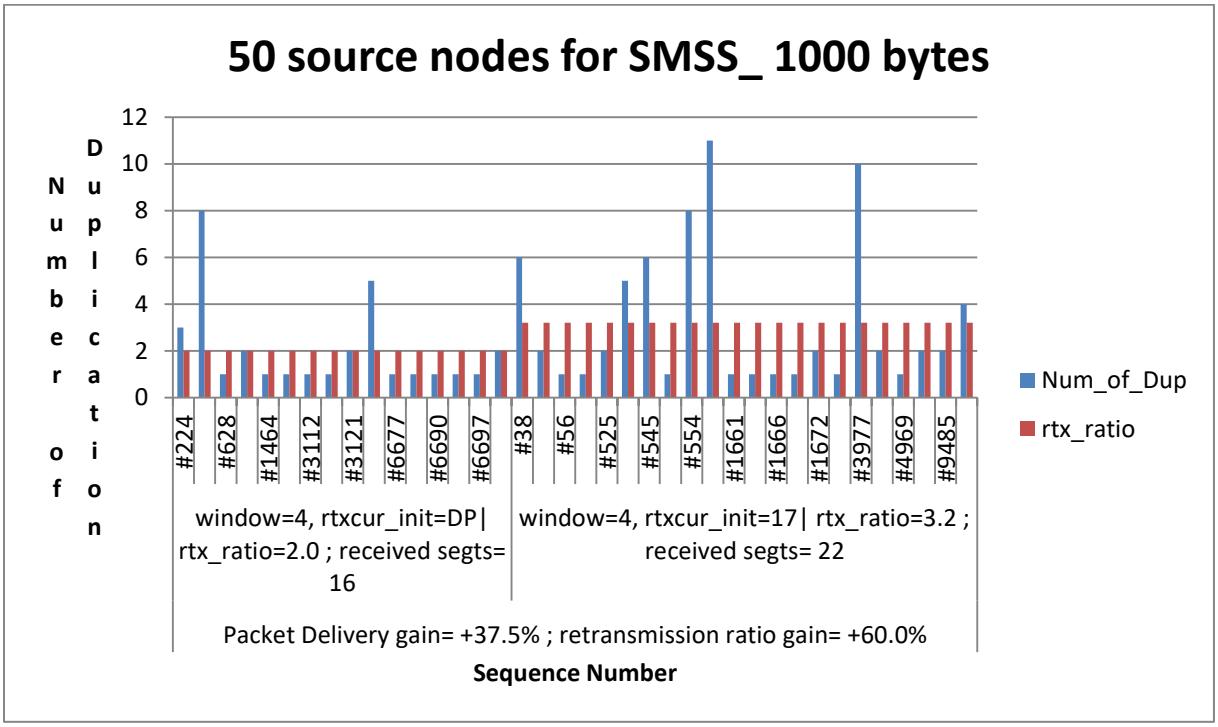


Figure 3.32 Impact of appropriate "rtxcur_init" value in scenario C while SMSS_ 1000 bytes

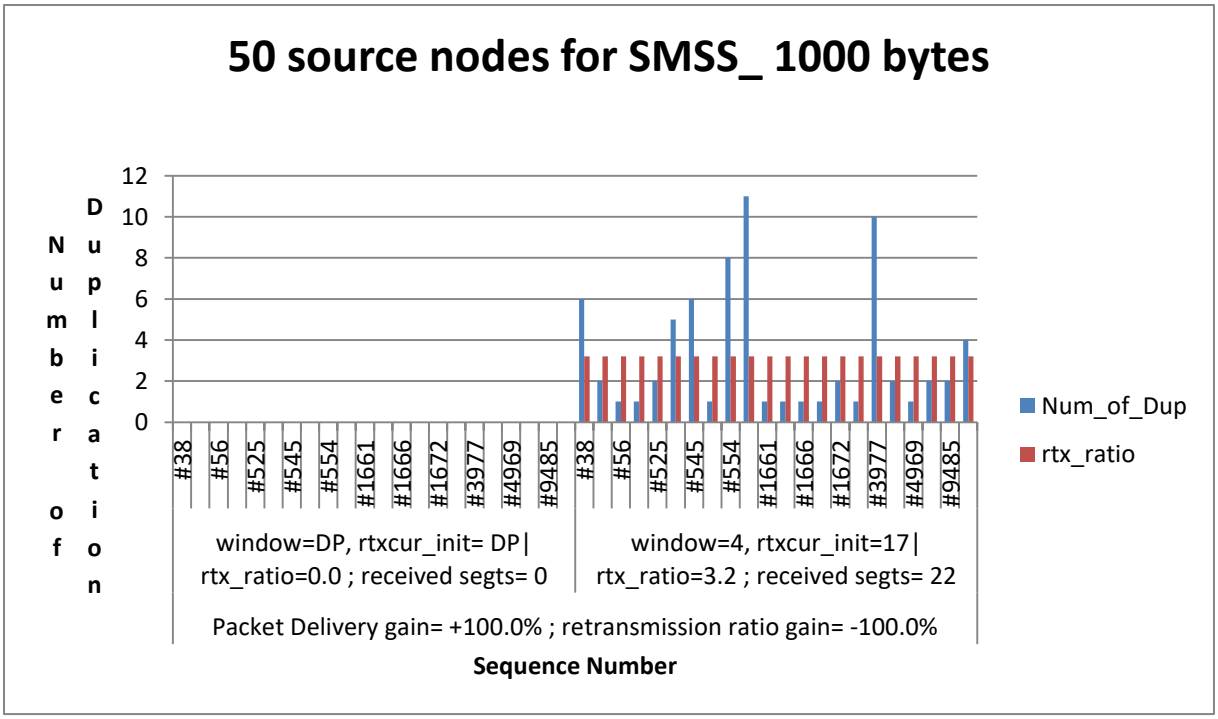


Figure 3.33 Illustration of TCP performance before and after both combined improvements in scenario C while SMSS_ 1000 bytes

Packet delivery

Figure 3.28 and *figure 3.21* show similarities with the 2500 bytes case (cf. *figure 4.16*) in scenario "B". Indeed, no reception is recorded during simulation before improvement. However, once an adequate size of the upper window limit size is found, we perceive packet delivery. There are 12 and 16 packets respectively for 1500 bytes and 1000 bytes cases. The 2500 bytes case shows a shift from 10 to 9 packets or -10% (cf. *figure 3.25*), which represents a lower disadvantage given the perceived gain when taking into account the other metric. The relatively small values of the maximum size of the congestion window for all cases show consistency. This could be the fact that this configuration, in addition to the disadvantageous characteristics of the channel, is conducive to more collision, interference, long queues caused by the volume of traffic generated. Thus, avoiding unnecessary packet loss could find its meaning in the use of congestion window limited to relatively small values.

In addition, we observe outstanding performances of TCP, when associating the impact of rtxcur_init to the first level of improvement. These are shifts from 9 to 19 or 111.1% (cf. *figure 3.26*), from 12 to 22 or +83.3% (cf. *figure 3.29*), from 16 to 22 or +37.5% (cf. *figure 3.32*) respectively for 2500, 1500 and 1000 bytes cases. It is also necessary to note the

consistency of the relatively high values of the rtxcur_init of 19 sec, 18 sec, and 17 sec respectively for 2500, 1500 and 1000 bytes cases. These could be explained by the presence of long queues at buffer caused by injected traffic.

Retransmission ratio

Only the 2500 bytes case shows improvements of -26.8% (cf. *figure 3.25*) and -16.7% (cf. *figure 3.26*) in both levels of improvement for a final retransmission rate of 2.5. For the other two cases, although major advances have been made in controlling the maximum size of the congestion window, *figures 3.29* and *3.32* show increases in the retransmission rate, respectively from 2.3 to 3.5 or +52.2% and from 2.0 to 3.2 or +60%.

The observation of a low retransmission rate in the case of 2500 bytes compared to the other two cases could translate a robustness of the packet either:

- By a better utilization of the bandwidth which is a scarce resource in the UWSN
- or by a certain immunity to the errors which can be a source of retransmission, when we know that the routing protocol DSDV would add a large amount of overheads and information packets that might be higher for the small size of the packet.

In the end, all cases show improvement of TCP performance by combining the impacts of both parameters. This is expressed by:

- ✓ Packets delivery passing from 10 to 19 or +90% for a retransmission rate in regression from 4.1 to 2.5 or -39% (cf. *figure 3.27*)
- ✓ Packets delivery passing from 0 to 22 for a retransmission rate of 3.5 (cf. *figure 3.30*)
- ✓ Packets delivery passing from 0 to 22 for a retransmission rate to 3.2 (cf. *figure 3.33*)

3.3.4 Summary of results regarding the segment size

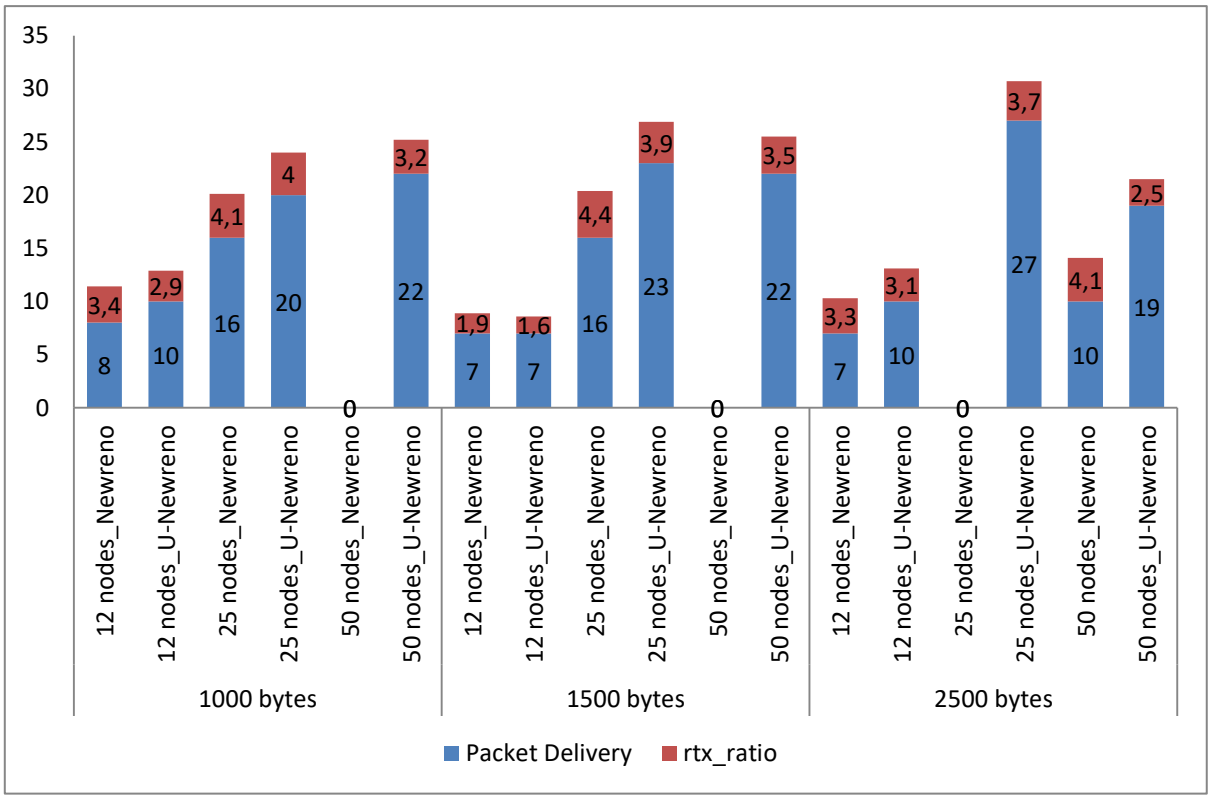


Figure 3.34 Summary of U-Newreno and Newreno performances relative to the SMSS size

In conclusion, taking into account the two metrics stated in paragraph 3.3.3.1, U-Newreno was able to improve Newreno performances. Indeed, all the cases of the different scenarios show an improvement (cf. *figure 3.34*). This would make it, an alternative to Newreno for a multi-hops UWSN communication.

Moreover, by observing the results of U-Newreno for each segment size, it is found that there is a relationship in terms of the number of received packets between 25 and 12 source nodes (cf. *figure 3.34*). In fact, this factor is:

- ✓ 2 for the case 1000 bytes (20 against 10)
- ✓ 3.3 for the case 1500 bytes (23 against 7)
- ✓ 2.7 for the case 2500 bytes (27 against 10)

On the other hand, the comparison of the number of packets received between 50 and 25 source nodes does not have this relation. In fact, except for the 1000 bytes case where the number of segments delivery is slightly greater (22 against 20), cases 1500 and 2500 bytes

show very poor results of 50 nodes compared to 25 source nodes namely 22 against 23 and 19 against 27 respectively.

This leads us to conclude that the number of source nodes has an impact on TCP performance in multihop UWSN communication. In addition to being able to determine the appropriate values of the maximum size of the congestion window and the rtxcur_init, it is wise to determine the optimum number of source nodes with respect to the coverage in terms of nodes for the area to be monitored. In our case, the optimal number is 25 nodes that represent the ¼ of the number of nodes involved in the communication.

4 Conclusion

The strong growth of underwater sensor networks has been accompanied by new challenges related to their physical and energy capacities. Indeed, because of their limited energy resources and their low transmitting power, radio channel disturbances seriously affect the performance of this type of network.

In this thesis, we presented a general view of a UWSN in order to understand the challenges of its realization. Subsequently, a presentation of the TCP including its variants was elaborated because it was the transport protocol concerned by our study. The proposed solution tends to improve the performances of TCP whose measurements were exclusively carried out by simulation under Aquasim.

Our solution U-Newreno for improving TCP performance in a UWSN is based on tuning the parameters of some TCP mechanisms. And these mechanisms can be used in such networks. The targeted mechanisms concerned congestion control and the calculation of the retransmission timer. These two mechanisms seemed important to us because they addressed significant challenges presented by this technology. Taking into account that UWSNs have a high rate of packet loss, we realized that it was necessary to allow the protocol to better handle the transition between slow-start and congestion avoidance phase to avoid a rapid growth of the sending window in case of successful packet delivery. This could be an important source of packet loss. The same is true for the second mechanism used, which is to allow a better estimation of the RTT timeout and thus relieve the network of the appearance of abusive RTO_S having as a direct consequence, retransmissions and consequently an energy load on the sensors. Although not deterministic as the method proposed in [48] where an algorithm to determine the optimal size of the packet to be used in a UWSN with multiple hops is provided, it has consisted of several simulations which allowed us to:

- control the upper limit of the congestion window,
- find an appropriate value of the 'rtxcur_init' used for initializing the RTT,

As a result, we noted significant improvements in the performance of TCP. The various tests described in section 3.3.3.2 have clearly illustrated the improvement of the behavior of TCP thanks to the chosen method. Concretely for the best-recorded performances, U-Newreno has reached improvements of packets delivery in overall with the most relevant ones as follows (cf. *figure 3.34*):

- ✓ going from 0 to 22 segments received case 1000 bytes
- ✓ going from 0 to 22 segments received case 1500 bytes

- ✓ going from 0 to 22 segments received case 2500 bytes

The same observation is made for the retransmission ratio of successfully arrived segments with reductions from (cf. *figure 3.34*):

- ✓ 3.4 to 2.9 or -14.7% cases 1000 bytes
- ✓ 4.4 to 3.9 or -15.8% cases 1500 bytes
- ✓ 4.1 to 2.5 or -39.0% cases 2500 bytes

However, despite our attempts to improve TCP performance, there are still a relatively large number of segments with a high retransmission ratio (cf. *figure 3.34*). This could be explained by two situations:

- either several dupACKs are transmitted by the destination before the sequence is received for the first time
- Either the RTT estimation is still not optimal so that several RTOs occur, demonstrating the difficulty of correctly setting the initial RTO in an underwater communication.

Thus, we can deduce that "Fast retransmit" and "Fast recovery" phases have been strongly solicited in such a communication and prove their efficiencies. We have also been able to demonstrate the relationship between the size of the segment (SMSS), the upper limit of the congestion window (CWND) and the estimation of the initial RTT. Our future work could focus on achieving TCP improvement in a mobile architecture. Alternatively, perform for another variant of TCP, improvements, and output a comparative study with Newreno.

Webography 1

[1] http://fr.atos.net/fr-fr/accueil/notre-profil.html

[2] http://fr.atos.net/fr-fr/accueil/nos-solutions.html

[3] https://www.slideshare.net/TalentedEurope/06-04-2017talented-europeatoscanariasv002

[4] http://fr.atos.net/fr-fr/accueil/notre-profil/profil-du-groupe/nos-marques.html#

[5] http://fr.atos.net/fr-fr/accueil/notre-profil/nos-clients.html

[6] http://fr.atos.net/fr-fr/accueil/notre-profil/partenariats.html

[10] http://zep333w0.distribution.edf.fr/wiki_iq/index.php/Accueil

[**12**] https://oi.erdf.fr/OI_MSI/pageaccueil.jsp

Bibliography 1

[7] Internal Integrator's manual v3.0 page 6

[8] Internal Integrator's manual v3.0 page 7

[9] Internal Integrator's manual v3.0 page 9

[11] Internal Integrator's manual v3.0 page 26

[13] Internal Integrator's manual v3.0 page 25

Webography 2

[39] http://netlab.ulusofona.pt/rc/book/3-transport/3_05/index.htm

[42] https://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf: "A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK, and Vegas"

[49] https://networksimulationtools.com/glomosim-simulator-projects, "Network simulators tools"

[51] http://www.irs.uji.es/uwsim/uwsim-features, "UWSim the UnderWater Simulator"

[52] https://www.isi.edu/nsnam/ns/, "The Network Simulator - ns-2"

[54] http://nile.wpi.edu/NS/, "OTcl: The User Language"

[55] http://www.mathcs.emory.edu/~cheung/Courses/558/Syllabus/A2-Tcl/OTcl.html, "OTcl: Object-Oriented Tcl"

[56] http://nile.wpi.edu/NS/analysis.html, "Trace Analysis example"

[58] http://dgt.dei.unipd.it/pages/read/58/, "Network Simulator 2 (NS2)"

[59] http://telecom.dei.unipd.it/ns/miracle/doxygen/index.html, "NS-MIRACLE Library: Multi InteRfAce Cross Layer Extension for NS"

[62] Federico Guerra, « World Ocean Simulation System (WOSS) Library", Version 1.5.0, http://telecom.dei.unipd.it/ns/woss/doxygen/ - federico@guerra-tlc.com

[66] http://obinet.engr.uconn.edu/wiki/index.php/Aqua-Sim

Bibliography 2

[1] Ian F. Akyildiz, Dario Pompili, Tommaso Melodia, "State-of-the-art in protocol research for underwater acoustic sensor networks", WUWNet '06 Proceedings of the 1st ACM international workshop on Underwater networks, Pages 7-16, September 25 - 25, 2006- ISBN:1-59593-484-7

[2] Jesús Llor and Manuel P. Malumbres, "Underwater Wireless Sensor Networks: How Do Acoustic Propagation Models Impact the Performance of Higher-Level Protocols?" Sensors, Volume 12, Issue 2 (February 2012), Pages 1312-1335

[3] Ian F. Akyildiz, Dario Pompili, Tommaso Melodia, "Underwater acoustic sensor networks: research challenges" Ad Hoc Networks Volume 3, Issue 3, May 2005, Pages 257-279

[4] Jennifer Trezzo, "Design and Implementation of an Adaptive Underwater Acoustic Modem and Test Platform**"**, ProQuest Dissertations Publishing, 2013, 1545450

[5] Mr. A. Manigopal, Mr. R. Panneerselvam M.E., "Underwater Wireless Sensor Networks: A Survey**",** IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 2, No.6, December 2012, ISSN: 2249-9555

[6] Manjula. R. B, Sunilkumar S. Manvi, "Issues in Underwater Acoustic Sensor Networks**"**, International Journal of Computer and Electrical Engineering, Vol.3, No.1, February 2011

[7] Emad Felemban, "Advanced Border Intrusion Detection and Surveillance Using Wireless Sensor Network Technology**",** Int. J. Communications, Network and System Sciences, 2013, 6, pp 251-259

[8] Mohsin Murad, Adil A. Sheikh, Muhammad Asif Manzoor, Emad Felemban, and Saad Qaisar, "A Survey on Current Underwater Acoustic Sensor Network Applications**",** International Journal of Computer Theory and Engineering, Vol. 7, No. 1, February 2015

[9] Christina Peach and Abdulrahman Yarali, "An Overview of Underwater Sensor Networks**"**, Proc. 9th International Conference Wireless Mobile Commun (ICWMC), 2013

[10] Tommaso Melodia, Hovannes Kulhandjian, Li-Chung Kuo, and Emrecan Demirors, "Advances In Underwater Acoustic Networking**",** Mobile Ad Hoc Networking: Cutting Edge Directions, Second Edition, The Institute of Electrical and Electronics Engineers, 2013

[11] Havard Austad, "Simulation of subsea communication network", Master thesis, 2014, http://urn.nb.no/URN:NBN:no-46449

[12] Ian F. Akyildiz, Dario Pompili, Tommaso Melodia, "Challenges for Efficient Communication in Underwater Acoustic Sensor Networks", Newsletter, ACM SIGBED Review - Special issue on embedded sensor networks and wireless computing, Volume 1, Issue 2, July 2004, Pages 3-8

[13] S. EL-Rabaie, D. Nabil, R. Mahmoud and Mohammed A. Alsharqawy, "Underwater Wireless Sensor Networks (UWSN), Architecture, Routing Protocols, Simulation and Modeling Tools, Localization, Security Issues and Some Novel Trends", Networking and Communication Engineering; Vol 7, No 8 (2015)

[14] Salvador Climent, Antonio Sanchez, Juan Vicente Capella, Nirvana Meratnia and Juan Jose Serrano, "Underwater Acoustic Wireless Sensor Networks: Advances and Future Trends in Physical, MAC and Routing Layers", 4, Issue 1 (2014), Pages 795-833, doi:10.3390/s140100795

[15] Milica Stojanovic and Pierre-Philippe Beaujean **"**Acoustic Communication**"** Springer Handbook of Ocean Engineering pp 359-386

[16] John G. Proakis, Joseph A. Rice, Ethem M Sozer, Malica Stojanovic, "Shallow water acoustic networks", IEEE Communications Magazine, Volume 39, Issue 11, November 2001, pp 114 – 119

[17] Tariq Ali, Low Tang Jung, Ibrahima Faye, "Classification of Routing Algorithms in Volatile Environment of Underwater Wireless Sensor Networks", International Journal of Communication Networks and Information Security (IJCNIS), Vol. 6, No. 2, August 2014

[18] Dario Pompili, "EFFICIENT COMMUNICATION PROTOCOLS FOR UNDERWATER ACOUSTIC SENSOR NETWORKS", ProQuest Dissertations Publishing, 2007. 3271578

[19] Reeta Mishra, "UW-ASNs: DESIGN CHALLENGES IN TRANSPORT LAYER", Assistant Professor, Department of Computer Science & Engineering, K .J. I.T, Gujarat, India, IJRET: International Journal of Research in Engineering and Technology, Volume 03 Special Issue 03, May-2014, NCRIET-2014

[20] Dario Pompili, Ian F. Akyildiz, "Overview of Networking Protocols for Underwater Wireless Communications", IEEE Communications Magazine, January 2009

[21] Peng Xie , Zhong Zhou , Zheng Peng, Jun-Hong Cui, Zhijie Shi, "SDRT: A reliable data transport protocol for underwater sensor networks", Ad Hoc Networks Volume 8, Issue 7, September 2010, Pages 708–722

[22] Rui Cao, Liuging Yang, "Reliable transport and storage protocol with fountain codes for underwater acoustic sensor networks", WUWNet '10 Proceedings of the Fifth ACM International Workshop on UnderWater Networks, Article No. 14, September 30 - October 01, 2010- ISBN: 978-1-4503-0402-3

[23] Dongseung Shin, Dongkyun Kim, "FRT: Fast and Reliable Transport Protocol for Underwater Wireless Sensor Networks", Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific

[24] Daniel Enrique Lucani, Muriel Médard, Milica Stojanovic, "Network coding schemes for underwater networks: the benefits of implicit acknowledgment", WuWNet '07 Proceedings of the second workshop on Underwater networks, September 14 - 14, 2007, Pages 25-32, ISBN: 978-1-59593-736-0

[25] Shaobin Cai, Nianmin Yao and Zhenguo Gao, "A reliable data transfer protocol based on twin paths and network coding for underwater acoustic sensor network", EURASIP Journal on Wireless Communications and Networking, December 2015, 2015:28

[26] Hao Wang, Shilian Wang, Eryang Zhang and Jianbin Zou, "A Network Coding Based Hybrid ARQ Protocol for Underwater Acoustic Sensor Networks", Sensors 2016, volume 16 Issue 9, p 1444; doi: 10.3390/s16091444

[27] Shaobin Cai, Zhenguo Gao, DeSen Yang, Nianmin Yao, "A network coding based protocol for reliable data transfer in underwater acoustic sensor", Ad Hoc Networks Volume 11, Issue 5, July 2013, Pages 1603-1609

[28] M. BanuPriya, V. HariPriya, A. Eniya, K. Deepika, "Challenges for Efficient Communication in Underwater Acoustic Sensor Network", On 4th September 2015 Organized by Department of CSE, ECE & EEE

[29] Goran Martinovic, Josip Balen, Drago Zagar, "A Cross-Layer approach and Performance Benchmarking in Wireless Sensor Networks", SENSIG'09/VIS'09/MATERIALS'09 Proceedings of the 2nd WSEAS International Conference on Sensors, and Signals and Visualization, Imaging and Simulation and Materials Science, November 07 - 09, 2009, pp 76 – 81, ISBN: 978-960-474-135-9

[30] Mohammad Abdur Razzaque, Simon Dobson, and Paddy Nixon, "Cross-Layer Architectures for Autonomic Communications", Journal of Network and Systems Management March 2007, Volume 15, Issue 1, pp 13–27

[31] Omar M. Sheikh and Samy A. Mahmoud "Cross-Layer Design for Smart Routing in Wireless Sensor Networks", Wireless Sensor Networks - Technology and Protocols, September 6, 2012, ISBN 978-953-51-0735-4, InTech, DOI: 10.5772/48605

[32] Pedram Vahdani Amoli, "An Overview on Researches on Underwater Sensor Networks: Applications, Current Challenges and Future Trends", International Journal of Electrical and Computer Engineering (IJECE) Vol. 6, No. 3, June 2016, pp. 955-962 ISSN: 2088-8708, DOI: 10.11591/ijece.v6i3.9201

[33] Jun-Hong Cui, Jiejun Kong, Mario Gerla, Shengli Zhou, "Challenges: Building Scalable and Distributed Underwater Wireless Sensor Networks (UWSNs) for Aquatic Applications", UCONN CSE Technical Report: UbiNet-TR05-02 September 2005

[34] Nor-Syahidatul N. Ismail, Liban Abdullahi Hussein, Sharifah H.S. Ariffin, "Analyzing The Performance of Acoustic Channel in Underwater Wireless Sensor Network(UWSN)", 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, 26-28 May 2010, DOI: 10.1109/AMS.2010.111

[35] Malumbres, M. P., Garrido, P. P., Calafate, C. T., & Oliver, J. (2009). **"**Underwater wireless networking technologies**"**, Encyclopedia of information science and technology (2nd Edition, pp. 3864–3958), Hershey, PA: IGI Publishing.

[36] Vishesh Mahajan, Harjit Singh, **"**A Review on Underwater Wireless Sensor Networks**",** GNDU RC Gurdaspur, Punjab, India, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 5, May 2015

[37] Kevin Downes, Merilee Ford, H. Kim Lew, Steve Spanier, Tim Stevenson, **"**Internetworking Technologies**",** Handbook, 2nd Edition Macmillan Technical Publishing, 201 West 103rd Street Indianapolis, IN 46290 USA, 1998 chapter 30.

[38] Douglas E. Comer, **"**Internetworking with TCP/IP, Volume I: Principles, Protocols, Architecture third Edition**"**, Departement of computer sciences Puredue University West Lafayette, IN 47907, 1995 chapter 13

[40] Allman, M., Paxson, V., and E. Blanton, **"**TCP Congestion Control**"**, RFC 5681, DOI 10.17487/RFC5681, September 2009

[41] Kevin Fall and Sally Floyd, **"**Simulation-based Comparisons of Tahoe, Reno, and SACK TCP**",** ACM SIGCOMM Computer Communication Review, Volume 26 Issue 3, July 1996, pages 5-21

[43] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, **"**The NewReno Modification to TCP's Fast Recovery Algorithm**"**, RFC 6582, DOI 10.17487/RFC6582, April 2012

[44] M. Mathis, J. Mahdavi, PSC, S. Floyd, LBNL, A. Romanow, Sun Microsystems, **"**TCP Selective Acknowledgment Options**"**, Network working group, RFC 2018, October 1996

[45] Lawrence S. Brakmo and Larry L. Peterson, **"**TCP Vegas: End to End Congestion Avoidance on a Global Internet**"**, IEEE Journal on Selected Areas in Communications, Volume 13, Issue 8, October 1995 , pages 1465 – 1480, ISSN: 0733-8716

[46] Naoya Iikubo, Masayuki Nakatsuka, Jiro Katto, Hayato Kondo, **"**Improving TCP Performance over Underwater Sensor Networks", the International Conference on UnderWater Networks and Systems (WUWNet), January 2008

[47] Muhammad Ayaz, Low Tang Jung, Azween Abdullah, Iftikhar Ahmad, **"**Reliable data deliveries using packet optimization in multi-hop underwater sensor networks", Journal of King Saud University – Computer and Information Sciences, Volume 24, Issue 1, January 2012, pages 41–48

[48] Eugenia Giancoli, Filippe Jabour, Aloysio Pedroza, "CTCP: Reliable Transport Control Protocol for Sensor Networks", Intelligent Sensors, Sensor Networks and Information Processing, International Conference, 15-18 December 2008, DOI: 10.1109/ISSNIP.2008.4762037

[50] Xiang Zeng, Rajive Bagrodia, Mario Gerla, "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks", Parallel and Distributed Simulation, PADS 98. Proceedings. Twelfth Workshop, 29-29 May 1998, DOI: 10.1109/PADS.1998.685281

[53] YOUSIF MOHSIN HASAN, "Communication and Computer Networks Simulator (NS2)", Article July 2014

[57] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Ya Xu, H. Yu "Network visualization with the VINT Network Animator NAM", Computer, Volume 33, Issue 11, November2000, Pages 63 – 68, ISSN: 0018-9162, DOI: 10.1109/2.881696

[60] Riccardo Masiero, Saiful Azad, Federico Favaro, Matteo Petrani, Giovanni Toso, Federico Guerra, Paolo Casari, Michele Zorzi "DESERT Underwater: an NS–Miracle-based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols", OCEANS, 21-24 May 2012, DOI: 10.1109/OCEANS-Yeosu.2012.6263524

[61] Paolo Casari, Cristiano Tapparello, Federico Guerra, Federico Favaro, Ivano Calabrese, Giovanni Toso, Saiful Azad, Riccardo Masiero, Michele Zorzi, "Open-source Suites for Underwater Networking: WOSS and DESERT Underwater", IEEE Network, Volume 28, Issue 5, September-October 2014, pages 38 –46, ISSN: 0890-8044, DOI**:** 10.1109/MNET.2014.6915438

[63] T. King, "A simulator for marine wireless sensor networks", Ph.D. thesis, Department of Computing Sciences, Texas A & M University, Corpus Christi, Texas, graduate project Spring 2011

[64] Peng Xie, Zhong Zhou, Zheng Peng, Hai Yan, Tiansi Hu, Jun-Hong Cui, Zhijie Shi, Yunsi Fei, Shengli Zhou, "Aqua-Sim: An NS-2 Based Simulator for Underwater Sensor Networks", OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges, 01 March 2010, ISSN: 0197-7385, DOI: 10.23919/OCEANS.2009.5422081

[65] UWSN Research group at the University of Connecticut, "User Tutorial for Aqua-Sim", Page 1-10

[67] V. Paxson, ACIRI, M. Allman, NASA GRC/BBN, "Computing TCP's Retransmission Timer", Network Working Group, November 2000, RFC 2988

Annex

The Tcl code of the simulation

###This script describes a 100 nodes topology of which 12 transmitting + one TCPSink

```
#==== General setting====
set opt(chan)           Channel/UnderwaterChannel
set opt(prop)           Propagation/UnderwaterPropagation
set opt(netif)          Phy/UnderwaterPhy
set opt(mac)            Mac/UnderwaterMac/BroadcastMac
set opt(ifq)            Queue/DropTail/PriQueue
set opt(ll)             LL
set opt(energy)         EnergyModel
set opt(txpower)        2.0
set opt(rxpower)        1.0
set opt(initialenergy)  10000
set opt(idlepower)      0.008
set opt(ant)            Antenna/OmniAntenna
set opt(ifqlen)         50      ;# max queue length in if
set opt(x)              2000    ;# X dimension of the topography
set opt(y)              2000  ;# Y dimension of the topography
set opt(z)              100
set opt(adhocRouting)   DSDV

# =====LINK LAYER SETTING=====
LL set mindelay_        50us
LL set delay_           25us
LL set bandwidth_       0       ;# not used

# =====QUEUE SETTING=====
Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# =====ANTENNA SETTING=====
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
#Antenna/OmniAntenna set Z_ 0.05
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# =====MAC LAYER SETTING=====
Mac/UnderwaterMac set bit_rate_  1.0e4 ;#10kbps
Mac/UnderwaterMac set encoding_efficiency_ 1
Mac/UnderwaterMac/BroadcastMac set packetheader_size_ 0 ;# #of bytes

# =====PHYSICAL LAYER SETTING=====
# Initialize the SharedMedia interface with parameters to make
```

```
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/UnderwaterPhy set CPThresh_  10  ;#10.0
Phy/UnderwaterPhy set CSThresh_  0    ;# sensing range of 200m in WSN
Phy/UnderwaterPhy set RXThresh_  0   ;# communication range of 200 in WSN
#Phy/WirelessPhy set Rb_ 2*1e6
Phy/UnderwaterPhy set Pt_  0.2818
Phy/UnderwaterPhy set freq_  25  ;# 25kHz
Phy/UnderwaterPhy set K_ 2.0    ;# spherical spreading

# =====TRANSPORT LAYER SETTING=====
Agent/TCP set packetSize_ 1500
Agent/TCP set windowInit_ 3
Agent/TCP set window_ 12
Agent/TCP set rtxcur_init_ 7

set ns [new Simulator]
set topo  [new Topography]
$topo load_cubicgrid $opt(x) $opt(y) $opt(z)

# Create a nam trace datafile.
set nf [open dsdv_nreno12_default.nam w]

# ----- Setup wireless environment. ----
set tracefd [open dsdv_nreno12_default.tr w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $nf $opt(x) $opt(y)

global TN
set TN 100
set num_node [expr $TN+1]
set god_ [create-god $num_node]
set chan_1_ [new $opt(chan)]
global defaultRNG
#$defaultRNG #seed $opt(seed)
#global node setting
$ns node-config -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop) \
                -phyType $opt(netif) \
                #-channelType $opt(chan) \
                -agentTrace ON \
        -routerTrace OFF \
        -macTrace OFF\
-topoInstance $topo\
-energyModel $opt(energy)\
        -txPower $opt(txpower)\
```

```
            -rxPower $opt(rxpower)\
            -initialEnergy $opt(initialenergy)\
            -idlePower $opt(idlepower)\
            -channel $chan_1_

# Create underwater nodes and set position.
#Setting node position block
for {set i 0} {$i < $TN} {incr i} {
        set node($i) [$ns node]
        $node($i) random-motion 0
        $ns initial_node_pos $node($i) 50.000000
}
        for {set j1 0} {$j1 < $TN} {incr j1 10} {
                incr x1 0
                incr y1 75
                $node($j1) set X_ $x1
                $node($j1) set Z_ $z1
                $node($j1) set Y_ $y1
                $ns at 0.000000 "$node($j1) setdest $x1 $y1 0.0"
        }
        for {set j2 1} {$j2 < $TN} {incr j2 10} {
                incr x2 0
                incr y2 75
                $node($j2) set X_ $x2
                $node($j2) set Y_ $y2
                $node($j2) set Z_ $z2
                $ns at 0.000000 "$node($j2) setdest $x2 $y2 0.0"
        }
        …       .
        …
        …

        for {set j10 9} {$j10 < $TN} {incr j10 10} {
                incr x10 0
                incr y10 75
                $node($j10) set X_ $x10
                $node($j10) set Y_ $y10
                $node($j10) set Z_ $z10
                $ns at 0.000000 "$node($j10) setdest $x10 $y10 0.0"
        }

# Set the node 100 as sink
        set node(100) [$ns node]
        $node(100) random-motion 0
        $ns initial_node_pos $node(100) 50.000000
        $node(100) set X_ 475
        $node(100) set Y_ 462
        $node(100) set Z_ 0
        $ns at 0.000000 "$node(100) setdest 475.0 462.0 0.0"
```

```
# Set TCP agent and the application
        for {set j 0} {$j < $TN} {incr j} {
        set tcp($j) [new Agent/TCP/Newreno]
        $ns attach-agent $node($j) $tcp($j)
        $tcp($j) set fid_ $j
        set cbr($j) [new Application/Traffic/CBR]
                $cbr($j) attach-agent $tcp($j)
                $cbr($j) set type_ CBR
                $cbr($j) set packet_size_ 65536
                $cbr($j) set rate 64kb
                $cbr($j) set random_ false
}

#Set TCPSink
        for {set k 0} {$k < $TN} {incr k} {
                set tcpSink($k) [new Agent/TCPSink]
                $ns attach-agent $node(100) $tcpSink($k)
        }

#Connect TCP traffic to the sink
        $ns connect $tcp(0) $tcpSink(0)
        $ns connect $tcp(90) $tcpSink(90)
        $ns connect $tcp(9) $tcpSink(9)
        $ns connect $tcp(99) $tcpSink(99)
        $ns connect $tcp(11) $tcpSink(11)
        $ns connect $tcp(71) $tcpSink(71)
        $ns connect $tcp(88) $tcpSink(88)
        $ns connect $tcp(28) $tcpSink(28)
        $ns connect $tcp(32) $tcpSink(32)
        $ns connect $tcp(62) $tcpSink(62)
        $ns connect $tcp(67) $tcpSink(67)
        $ns connect $tcp(37) $tcpSink(37)

#Schedule the simulation
        $ns at 0.5000000 "$cbr(0) start"
        $ns at 0.5000000 "$cbr(90) start"
        $ns at 0.5000000 "$cbr(9) start"
        $ns at 0.5000000 "$cbr(99) start"
        $ns at 0.5000000 "$cbr(11) start"
        $ns at 0.5000000 "$cbr(71) start"
        $ns at 0.5000000 "$cbr(88) start"
        $ns at 0.5000000 "$cbr(28) start"
        $ns at 0.5000000 "$cbr(32) start"
        $ns at 0.5000000 "$cbr(62) start"
        $ns at 0.5000000 "$cbr(67) start"
        $ns at 0.5000000 "$cbr(37) start"
        $ns at 295.000000 "$cbr(0) stop"
        $ns at 295.000000 "$cbr(90) stop"
        $ns at 295.000000 "$cbr(9) stop"
        $ns at 295.000000 "$cbr(99) stop"
```

```
        $ns at 295.000000 "$cbr(11) stop"
        $ns at 295.000000 "$cbr(71) stop"
        $ns at 295.000000 "$cbr(88) stop"
        $ns at 295.000000 "$cbr(28) stop"
        $ns at 295.000000 "$cbr(32) stop"
        $ns at 295.000000 "$cbr(62) stop"
        $ns at 295.000000 "$cbr(67) stop"
        $ns at 295.000000 "$cbr(37) stop"

# Run the simulation
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam dsdv_nreno12_default.nam &
        exit 0
        }
#call the finish procedure after 5 miniutes of simulated time
$ns at 300 "finish"
$ns run
```