Network traffic characterisation, analysis, modelling and simulation for networked virtual environments

Juan Luis Font Calvo

July 2019

Departmento de Arquitectura y Tecnología de Computadores
Escuela Técnica Superior de Ingeniería Informática

Doctoral dissertation

Network traffic characterisation, analysis, modelling and simulation for networked virtual environments

Juan Luis Font Calvo

Supervisor:
José Luis Sevillano Ramos

July 2019

Juan Luis Font Calvo
Network traffic characterisation, analysis, modelling and simulation for networked virtual environments
Doctoral dissertation, July 2019
Supervisor: José Luis Sevillano Ramos

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática
Departmento de Arquitectura y Tecnología de Computadores
Avd. Reina Mercedes S/N
41700 Sevilla

Abstract

Networked virtual environment (NVE) refers to a distributed software system where a simulation, also known as *virtual world*, is shared over a data network between several users that can interact with each other and the simulation in real-time. NVE systems are omnipresent in the present globally interconnected world, from entertainment industry, where they are one of the foundations for many video games, to *pervasive games* that focus on e-learning, e-training or social studies. From this relevance derives the interest in better understanding the nature and internal dynamics of the network traffic that vertebrates these systems, useful in fields such as network infrastructure optimisation or the study of *Quality of Service* and *Quality of Experience* related to NVE-based services. The goal of the present work is to deepen into this understanding of NVE network traffic by helping to build network traffic models that accurately describe it and can be used as foundations for tools to assist in some of the research fields enumerated before.

First contribution of the present work is a formal characterisation for NVE systems, which provides a tool to determine which systems can be considered as NVE. Based on this characterisation it has been possible to identify numerous systems, such as several video games, that qualify as NVE and have an important associated literature focused on network traffic analysis. The next contribution has been the study of this existing literature from a NVE perspective and the proposal of an *analysis pipeline*, a structured collection of processes and techniques to define *microscale* network models for NVE traffic. This analysis pipeline has been tested and validated against a study case focused on *Open Wonderland (OWL)*, a framework to build NVE systems of different purpose. The analysis pipeline helped to defined network models from experimental OWL traffic and assessed on their accuracy from a statistical perspective. The last contribution has been the design and implementation of simulation tools based on the above OWL models and the network simulation framework *ns-3*. The purpose of these simulations was to confirm the validity of the OWL models and the analysis pipeline, as well as providing potential tools to support studies related to NVE network traffic. As a result of this final contribution, it has been proposed to exploit the parallelisation potential of these simulations through *High Throughput Computing* techniques and tools, aimed to coordinate massively parallel computing workloads over distributed resources.

Resumen

Un *entorno virtual en red* (*networked virtual environment, NVE)* hace referencia a un tipo de software distribuido donde una simulación, también denominada *mundo virtual*, es compartida a través de una red de datos entre varios usuarios que pueden interactuar entre ellos y la propia simulación en tiempo real. Los sistemas NVE son omnipresentes en el mundo actual, global e interconectado, desde la industria del entretenimiento, donde son uno de los fundamentos sobre los que se asientan muchos video juegos, a los *juegos ubicuos* centrados en e-learning, e-training o estudios sociales. Debido a esta relevancia se deriva el interés en comprender mejor la naturaleza y dinámicas internas del tráfico de red que vertebra estos sistemas, de utilidad en campos como la optimización de infraestructuras de red o el estudio de *Quality of Service* y *Quality of Experience* asociados a servicios basados en sistemas NVE. El presente trabajo pretende profundizar en esta comprensión del trafico de red generado por sistemas NVE mediante la definición de modelos de tráfico de red que lo describan de forma precisa y que puedan ser usados como base para herramientas útiles para los campos de estudio mencionados anteriormente.

La primera contribución del presente trabajo es una caracterización formal para sistemas NVE, la cual proporciona una herramienta para determinar que sistemas pueden considerarse NVE. En base a esta caracterización ha sido posible identificar numerosos sistemas, como por ejemplo varios videojuegos, que cualifican como NVE y tienen una importante literatura asociada centrada en el análisis de tráfico de red. La siguiente contribución ha sido el estudio de esta literatura desde la perspectiva de los sistemas NVE y la propuesta de un *proceso de análisis*(*analysis pipeline*), un conjunto estructurado de procesos y técnicas para definir modelos a nivel de *micro-escala* para el tráfico de red de sistemas NVE. Este proceso de análisis ha sido probado y validado contra un caso de estudio centrado en *Open Wonderland (OWL)*, una herramienta (*framework*) para construir sistemas NVE de diverso propósito. El proceso de análisis ha ayudado a definir modelos de red partiendo de tráfico experimental de OWL y a evaluar su precisión desde el punto de vista estadístico. La última de las contribuciones ha sido el diseño e implementación de herramientas de simulación basadas en los anteriores modelos para OWL y la herramienta de simulación de redes *ns-3*. El objetivo de estas simulaciones ha sido confirmar la validez de los modelos de OWL y el proceso de análisis, a la vez que proporcionar

herramientas potencialmente útiles en las áreas de estudio relacionadas con el tráfico de sistemas NVE. Como resultado de esta última contribución se ha propuesto explotar el potencial de paralelización de estas simulaciones a través de técnicas y herramientas de *computación de alto rendimiento*, (*High-Throughput computing*), cuyo objetivo es coordinar tareas de cómputo masivamente paralelas sobre recursos distribuidos.

Agradecimientos

La realización de esta tesis doctoral ha sido un largo viaje que ha abarcado años, todo tipo de eventos vitales y giros de guión. Me alegra poder agradecer por fin a todos aquellos que son casi tan responsables como yo de que haya sido posible.

En primer lugar agradecer a mi director de tesis José Luis Sevillano por todos sus esfuerzos, ayuda y confianza en que esta tesis podía materializarse, a Christian Callegari del Dipartimento di Ingegneria dell'Informazione de la Università di Pisa por hospitalidad y ayuda durante mi estancia en Italia, y a mis antiguos compañeros del Departamento de Arquitectura y Tecnología de Computadores Daniel Cascado por sus esfuerzos y tutela, y a Rosa Yañez por la ayuda, cordura y amistad aportadas a la causa.

En segundo lugar, agradecer a mis padres Concepción y Juan Luis, por apoyarme incondicionalmente, no sólo durante este proceso, sino durante toda mi vida. Mamá, Papá, os quiero.

En tercer lugar, agradecer a todos esos amigos incondicionales repartidos por el camino entre Los Palacios (y Villafranca) y Ámsterdam con escala en Pisa y otros lugares igualmente interesantes. Sois geniales.

Acknowledgement

The making of this PhD has been a journey involving many years, all kind of vital events and plot twists. I am glad to finally being able to thank all those who are almost as responsible as myself for all this to happen.

First, thanks to my supervisor José Luis Sevillano for all his efforts, help and confidence in this project to actually happen, to Christian Callegari from the Dipartimento di Ingegneria dell'Informazione of the Università di Pisa for his help and hospitality during my stay in Italy, and to my former colleagues from Departamento de Arquitectura y Tecnología de Computadores Daniel Cascado for his efforts and tutelage, and Rosa Yañez for the help, sanity and friendship she contributed to the cause.

Second, thanks to my parents Concepción and Juan Luis, for their unconditional support and love, not only during all this process, but also during my whole life. Mom, Dad, I love you.

Third, thanks to those unconditional friends scattered all over the way between Los Palacios (y Villafranca) and Amsterdam, with stopovers in Pisa and other equally interesting places. You all are great.

Quack!

Contents

Introduction

1

With magic, you can turn a frog into a prince. With science, you can turn a frog into a PhD and you still have the frog you started with.

— Terry Pratchett, *The Science of Discworld*

Nowadays the concept of *virtual world* has become familiar to wider audiences thanks to video games such as *Second Life*, *World of Warcraft* or *Minecraft*. Video games are not the only systems based on this paradigm, which has been used since the 1980s in other fields such a e-learning, e-training or social research. Although their focuses range from entertainment to learning or just plain social interaction, all these systems base their user experience on the concept of virtual world, a distributed simulation shared by several users over a data network where users can interact between them and the virtual world itself. In this context, a networked virtual environment (NVE) can be broadly defined as "a software system in which multiple users interact with each other in real-time, even though those users may be located around the world" [SZ99], [Chu+01].

The ultimate goal of the present work is to deepen into the understanding of the network traffic dynamics that articulate a NVE. For this purpose a characterisation for NVE systems is proposed, providing a tool that will help to identify compliant systems and approach the existing literature on their network traffic analysis from a NVE perspective. From this new lecture of the existing bibliography, analysis and modelling procedures focused on NVE network traffic will be proposed. This work also include a study case based on a chosen NVE system and the results from applying the proposed analysis and modelling procedures, and the deliverables in the form of valid network traffic models and simulations based on them.

1.1 Historical overview

To better understand the significance of the research and study associated to NVE systems it is necessary to make a brief overview of their genesis, evolution and use in different contexts, ranging from military to academic and scientific purposes, as well as been an important part of a multibillion global business as it is that of video games. The following sections will provide a historical overview focused on these different fields, referencing some of the most significant milestones and their significance from the perspective of the NVEs.

1.1.1 Military use

NVE systems were initially sought as cost effective alternative training with real combat vehicles and equipment while avoiding all the implicit risks associated to such an activity. Thus, military organisation have sponsored and hosted research projects to produce training technologies based on NVE systems [Rhe91], [Arm].

SIMNET “was the first successful implementation of large-scale, real-time, man-in-the-loop simulator networking for team training and mission rehearsal in military operations” [MT95]. The SIMNET research project was developed for the DARPA between 1983 and 1990, it provided a real-time distributed simulator for combat simulation. SIMNET had the features that characterize a NVE: it was designed to support hundreds of concurrent users using network technology initially based on 56 Kbit/s dial-up lines and moving to faster T-1 links in later stages of the project. Users were accommodated in the simulation stations, which provided an on-screen graphical representation of the virtual world as well as the required peripherals to interact with it. The simulation station was represented by a vehicle within the virtual world capable of hosting several human crew members. Audio generated by crew members was also transmitted within the virtual world.

The Distributed Interactive Simulation (DIS) is an IEEE standard [Dis12] for conducting real-time platform-level wargaming across multiple host computers. DIS started as a follow-up project of SIMNET aimed to document its communication protocol, describing a fully distributed architecture for heterogeneous clients scalable up to 500 users.

Another descendant of the SIMNET project was the Mounted Warfare TestBed (MWTB), located at Fort Knox, Kentucky, started in 1986 as the initial site for the *SIMNET-D* [Gar89] program, a spin-off of the original SIMNET project. The MWTB was the premier site for distributed simulation experiments in the US Army for over 20 years. It used simulation systems to perform experiments that examined current and future weapon systems, concepts, and tactics. Some of the technologies developed within the MWTB include the *ACRT* virtual simulator [CFM94] and the *Virtusphere*, a device that allows the user to walk around the simulated battlefield.

SIMNET also founded the basis of several companies and professional careers that transfer part of the knowledge and technology developed during the project to other non-military fields. Examples of this process are *MetaVR Inc.* (W. Garth Smith) [Met] and *MaK Technologies* [MAK] (Warren Katz and John Morrison), companies that still provide simulation and training software and resources for civil and engineering usage; *Zipper Interactive* founded by Brian Soderberg [Int12], a defunct video game company part of *Sony Computer Entertainment*, responsible of the development of the *SOCOM* (U.S. Special Operations Command) peer-to-peer (P2P) game series; or *Wiz!Bang*, a video game development company founded by Drew Johnston [Mob], who has developed a lengthy career in the video game industry working for the *Microsoft Windows Gaming Platform* team, among others.

In recent years, this technology transfer has worked in the opposite direction, where well established technologies from the gaming consumer computer industry have found their way into military training programs and *serious games*, games not focused on entertainment purposes, but training and education. One example is the *DARWARS* [Vot04] research program started in 2003, intended to develop and deploy military training systems based on low-cost, mobile, web-centric, simulation-based, *lightweight* systems taking advantage of the ubiquitous presence of the PC, multi-player games, virtual worlds, off-the-shelf PC simulations, intelligent agents, and online communities. Another good example of technology transfer are the Virtual Battlefield Systems 1 (VBS1) released in 2002, and its successors *VBS2* released in 2007 and *VBS3* released 2014 [Int], all of them military simulators that heavily rely on modern gaming technology and generally considered as *serious games*. VBS1 is based on the first-person shooter (FPS) video game *Operation Flashpoint* developed by *Bohemia Interactive Australia*. The system allows training in military tactics in an interactive multiplayer 3D world. This combination of military simulator functionality and modern gaming technology proved to be a success and resulted in a broad military customer base.

Besides training, NVE technologies and system have been also used as strategic communication tools. The first usage of gaming technologies in support of U.S. Army recruiting and strategic communication has been *America's Army* [Arm], a platform to develop first-person shooter games released in 2002 by the U.S. Army, based in the Unreal game Engine and designed to allow young Americans to virtually explore the Army and helping them to determine if soldering matches their expectations. An evolution of America's Army is the Virtual Army Experience (VAE) [Rel07], a mobile U.S. Army simulator created in 2007 by the Army development team which brought the army's computer game, *America's Army: Special Forces (Overmatch)* to a life-size, networked environment to provide visitors with an experience of soldiering.

1.1.2 Entertainment industry and video games

A video game is defined as an electronic game that involves human interaction with an user interface to generate visual feedback on a video device (TV, computer monitor, etc.) [BRH72]. The main goal of video games is providing a rewarding experience to the player. This reward does not only lie in the final goals of the game but also the process itself to achieve them [MSB13].

Video games have evolved from a very focused market in mid 1970s to important industry (sometimes referred as interactive entertainment industry) billing more than $80 billions [BV14] worldwide in 2014 and jumping up to $138 billions in 2018 [BV18]. This growth has sustained the evolution and proliferation of numerous genres [Wol02], game mechanics and topics that currently conform a rich gaming ecosystem. In a classification attending technical features, video games relying on networking capabilities define the online games category, those played over a data network [Ada10]. These games have become ubiquitous thanks to the universalization of the Internet.

Early NVE-compliant video games

As discussed in section 1.1.1, NVE systems appeared in the context of the military context as a simulation and instruction tool. However, there are many video game genres and game dynamics based on the NVE concept (see Section 2.1 for a formal characterisation of NVE). Thus, there is a significant overlapping between the NVE paradigm and the characteristics of online video games: both consist in electronic, specifically digital , systems where several users engage in a virtual world simulation shared over a data network and can interact between them in real-time. Thus, online games which dynamics involve the recreation of a virtual world shared over a data network where users can interact and undertake challenges, are de-facto NVE systems designed to provide an entertaining experience.

Some of the milestones in the convergence of video games into true NVE systems are linked to the technical advances making possible the incorporation of the defining characteristics of the NVEs into the consumer video game world. In the canonical accounts of the history of video games, *Maze War* (1974) is regarded as one of the earliest examples of FPS, as well as inventing or disseminating several important concepts that are used by many of the video games to follow [RP09] [Ars09]. Gameplay consisted in players roaming around a maze, other players are represented by eyeballs, when the player encounters another players in the maze, the player can gain or lose points by shoot or been shot during the match. There are several pioneering features in Maze Wars that are closely linked to the NVE concept:

- Initially developed to run on the Imlac PDS-1, a popular graphical display system which provided input and output peripherals.
- First-person 3D perspective, players are provided with a visual representation of the maze (their shared virtual world) from their point of view.
- Avatars, players are represented by eyeballs. Previous games usually represented players as spaceships and other kind of vehicles. This is one of the earliest examples where players are represented by *organic beings*.
- Network play, an early example of P2P computers, unlike previous games played within the same minicomputer or mainframe using different terminals.
- Client-server networked play, this feature achieved in later improved versions allowed to play the game over ARPANET in 1977, making it one of the earliest examples of Internet gaming.

Moria [s15], [Add] (1975), an early role-playing game (RPG) game, borrowed concepts from Maze War. It was developed to run on the Programmed Logic for Automatic Teaching Operations (PLATO) system [SE79], the first generalized computer assisted instruction system. Moria took advantage of the many multi-user and networking features of PLATO to define a gameplay where parties of up to ten players to travel as a group, while exploring dungeons and featuring a wireframe first-person perspective display. An feature that makes Moria to further converge towards the NVE definition is the ability of players to communicate among them using text messages during the gameplay, adding a social factor.

MUD1 (1978), initiator of the so-called Multi-User Dungeon (MUD) genre, was initially developed on a DEC PDP-10 at Essex University. MUD1 a multiplayer game where users interact in real-time in a *Dungeon & Dragons* inspired virtual world. MUD1 is text based, so the user interacts with the system typing combination of commands and keywords and receives a description of the virtual world and its events in text format. MUD1 can be considered a milestone in the convergence of video games into NVEs because it became the first online Internet role-playing game once the Essex University connected its internal network to ARPANet [MP03]. Moreover, it is considered one of the virtual world game still in existence, being playable via telnet protocol [Tot].

Evolution and popularisation

After early games defined and most of the key elements found in NVE systems, later titles pushed forward these features, improving the graphic representation of the virtual world, pushing the boundaries of connectivity to the range of local area networks (LANs) or popularizing among large audiences genres build around the NVE concept.

Dogfight [SGI93] (1985) was developed by Silicon Graphics, Inc. (SGI) as a demo program for their workstations. The game is a multiplayer flight simulator where the player has a first person view of a cockpit, other users in the game are represented by planes which play the role of avatars. The game is considered a technical landmark because of its 3D render capabilities and especially networking features, being the first game to use the TCP/IP protocol.

Games pioneering in technical features that would define the NVE experience (graphic representation of the virtual world, avatar, networking) gave way during mid and late 80s to games that contributed to the popularisation and expansion of different NVE-related genres, especially the FPS and the massively multiplayer online role-playing game (MMORPG) ones. Both genres will prove increasingly popular during the whole decade of 1990s and booming in 2000s with the ubiquity of Internet and improvements in network technologies.

Habitat (1985) developed by *Lucasfilm Games Group*, is another milestone in the video game convergence towards the NVE paradigm. The game has been defined in [MF91] as a ““multi-player online virtual environment”, a cyberspace. Each player uses his or her home computer as a front end, communicating over a commercial packet-switching data network to a centralized back end system. The front end provides the user interface, generating a real-time animated display of what is going on and translating input from the player into requests to the back end. The back end maintains the world model, enforcing the rules and keeping each player’s front end informed about the constantly changing state of the universe”. The back end enables the players to interact not only with the world but with each other. The game was available for the Commodore 64 platform, one the most widespread 8-bit home computer of the 1980s, relying on the online service Quantum Link, the corporate progenitor of America Online (AOL). These factors made Habitat the first attempt at a large-scale commercial virtual community that was graphically based [Rob94]. Attending to the Habitat’s description provided in [MF08] and [MF91], it can be determined that includes most of the defining NVE characteristics.

Neverwinter Nights (1991) was the first MMORPG to display graphics [Bai04]. The gameplay is divided between a screen providing information about the game party members, current status and geographical location within the virtual world and the screen for combat mode where each character is represented by an avatar and battles take place. During its lifespan running on AOL network from 1991 to 1997, the capacity of the Neverwinter Nights servers grew from 50 to up to 500 concurrent users per server. Near the end of its run in 1997, the game had 115 000 subscribed players and an average of 2000 concurrent players during peak hours. Much of the game's popularity was based on the presence of active and creative player guilds, who staged many special gaming events online for their members.

Doom [Sof93] (1993), a science fiction FPS video game by *id Software*, is considered one of the most significant and influential titles in the video game industry, responsible of ushering the mainstream popularity of the FPS genre. While Maze War was innovative by defining and shaping many video game concepts, Doom helped to bring into wide audiences many video game concepts and features that are current staples in the genre. Among others, it provided immersive 3D graphics, support for customized additions and content via data files known as *WADs* (ludicrous acronym for *Where's All the Data?*) and networking capabilities allowing different online multiplayer modalities on both LAN and the Internet [Kus04].

Internet and the global scale of NVE-based games

The Internet explosion during the later half of the 1990s greatly expanded the target audience for all kind of online games during that period and the next decade of the 2000s. The shared virtual worlds expanded from LANs to the Internet, breaking the barrier of locality and allowing interactions over a global and much more heterogeneous data network. Some of the genres that gained great popularity was MMORPG, a combination of classic RPG and multiplayer online games where players interact between them in a virtual world. Titles such as *Ultima Online* or *EverQuest* were among the first to popularize the genre, while the massive commercial success arrived in 2004 with World of Warcraft (WoW) by *Blizzard Entertainment*, having 11 million subscribers at its peak in 2010 [Ent10]. MMORPGs games in general and WoW in particular not only become a powerhouse in the video game industry, but also sparked academic interest. With an audience of millions of players all around the world a revenue in the range of billions of dollars, studying the relationship between network quality of service (QoS) and quality of experience (QoE) inspired studies such as [FRS05], [Che+06b] and [RSR08]. The technical challenges posed by such a number of concurrent players all around the world was inspired studies such as [PKK05], [Kim+05], [Che+05], [Che+06a] and [SKR07], also addressing network traffic optimisation in [Sal+12]. Another interesting aspect of the relationship of MMORPG games and academia has been their usefulness as tools for other disciplines such as epidemiology, as shown in [Bal07] and [GG15].

The FPS genre based on online multiplayer mechanics also flourished during this period, with sagas such as *Counter-Strike* (2000-present,*Valve Corporation*), *Halo* (2001-present, published by *Xbox Game Studios*), *Quake* series (1996-present, *id Software*), *Call of Duty* (2003-present, distributed by *Activision*), *PlayerUnknown's Battlegrounds* (2017, *PUBG Corporation*) or *2016, Overwatch* (*Blizzard Entertainment*) becoming extremely popular and contribution to the raise in popularity of the *Esports*, a form of competition using video games [HS17]. But while MMORPGs involved challenges about scaling up to millions of concurrent users and synchronisation of the virtual worlds across many servers, online FPS games posed technical challenges related to tight synchronisation constraints. Modelling their network traffic was addressed in studies such as [Ami+13], [CB07] focused on the network traffic modelling, or [Fen+02], which used network modelling to help with the provisioning of network infrastructures to successfully accommodate large number of concurrent and distributed users. QoE-related studies such as [Wat+06], dealt with player's perceived quality, others such as [ZA04] focused on the impact of network QoS on the user's QoE. The study performed in [LC17] goes further in problematics tightly associated to the strict synchronisation requirements of FPS games, such as the so-called "shot around the corner", where synchronisation inconsistencies can lead to the kill of a player even if it is properly covered.

During the decade of 2010s, the predominance of online multiplayer titles even pushed popular *open world* game series, focused on offline single-player dynamics where the player can freely explore a virtual world, to expand their features or release titles supporting online multiplayer mode. Some of these examples are *Fallout 76* (2018, *Bethesda Game Studios*), *Grand Theft Auto Online* (2013, *Rockstar Games*) or *Red Dead* series (2010 and 2018, *Rockstar Games*). This trend reflects the popularity of the networked virtual world paradigm in the game industry and the economical weight associated to it. Due to the involved game dynamics, the above titles combine the technical challenges of huge open world scenarios and multiple synchronised server of the MMORPG games with the tight timing constrains and real-time synchronisation of FPS titles, posing more questions for the fields of study around NVE network traffic and providing new tools for others ([MH12], [FLK17]).

1.1.3 Academia and research

NVEs have been associated to research from their early inception, sometimes as a canvas where to build and perform experiments, sometimes as subject of study themselves. One of the earliest academic NVE was *NPSNET* [SZ99], developed in 1986 by the Department of Defense (DoD) of the U.S. as an evolution of the previously mentioned *SIMNET*, focused on implementing large-scale virtual environments (LSVEs) at LAN network level. The decade of 1990s witnessed the emergence of several more research NVE. Distributed Interactive Virtual Environment (DIVE), released in 1991 by the Swedish Institute of Computer Science, was an Internet-based multi-user virtual reality (VR) system where participants navigate in 3D space where they can interact with other users and applications [Hag96]. Another example of academic NVE was *MASSIVE-1* [GB95], developed by the Computer Research Group at the University of Nottingham. It was a distributed multi-user VR system running on high end Sun Microsystem and SGI platforms. It had textual, graphical and audio client programs. Later *MASSIVE-2* and *MASSIVE-3* improved scalability and moved to a TCP-based *client-server* model.

The initial trend in the development of NVE platforms was to design and build specific-purpose systems with specific research questions in minds. For example in [YK13] the authors provide a comprehensive list of NVEs systems which use different P2P techniques for virtual world synchronisation, many of them just build as a proof of concept or to study a very specific feature instead of trying to provide a complete virtual world experience to end users. The increase in complexity of the NVE systems together with the advances in computing power, 3D graphics and networking experimented during the decade of 1990s, propitiated the emergence of new generic NVE platforms. The *openness* of these systems, in both license and technical terms, made them attractive frameworks for scientific and researchers that started to use them as a canvas.

Specialised NVE frameworks

The initial trend in the development of NVE platforms was to design and build specific-purpose systems with specific research questions in minds. For example in [YK13] the authors provide a comprehensive list of NVEs systems which use different P2P techniques for virtual world synchronisation, many of them just build as a proof of concept or to study a very specific feature instead of trying to provide a complete virtual world experience to end users. The increase in complexity of the NVE systems together with the advances in computing power, 3D graphics and networking experimented during the decade of 1990s, propitiated the emergence

of new generic NVE platforms. The *openness* of these systems, in both license and technical terms, made them attractive frameworks for scientific and researchers that started to use them as a canvas.

Open Cobalt is a free and open-source (MIT license) multi-platform software for constructing, accessing, and sharing virtual worlds (*virtual workspaces* in the terminology of the project) both on local area networks or across the Internet, with no need for centralized servers. Developed at Duke University and initially released in 2007, it is based on Squeak, a dialect of Smalltak, including features such as integrated web-browsing, voice and text chat, and other embedded applications accessible in-game. Open Cobalt aims to create different virtual worlds hyperlinked between them thus constituting a *metaverse*. Most of the studies performed using Open Cobalt as experimental platform are focused on e-learning and e-training ([FB14], [TL14]). As of 2019, the Open Cobalt project is currently dormant.

The *OpenSimulator* project [Fis09], sometimes referred as just *OpenSim*, was founded in January 2007 by Darren Guard with the goal of providing an open-source 3D NVE server that could become a standard framework to build NVEs. Initially based on the open-source *Second Life* client and messaging protocols, it currently provides a multi-platform, multi-user 3D application server. OpenSimulator allows a high degree of flexibility to virtual world developers to customize their worlds thanks to its extensibility. Its core is written in C# under a BSD License, a commercially friendly license to embed OpenSimulator in products. Academic studies based on or using OpenSimulator range from e-learning topics ([Vil+10]), to archaeology ([SM13]), robotics ([JEP10]) or network scalability of NVE systems ([LH13]).

Another noteworthy mention of general purpose NVE with strong rooting in academia is Open Wonderland (OWL), a Java open-source toolkit for creating collaborative 3D virtual worlds. OWL will be subject of study in the present work as a study case of NVE system. Further details about OWL can be found in Section 2.2.

Commercial video games as academic NVE frameworks

The second half of the 2000s saw a new trend where popular video games that qualified as NVE started to be extended and used for academic and research purposes. Using commercial video games as platform to build research NVE systems is not a new trend. Projects such as *Colyseus* ([BPS06] and [BAS13]), a NVE platform based on P2P synchronisation techniques based on *Quake II* to study network scalability and load balancing, or the use of *BrowserQuest* as platform for health-related projects such as [GG15] are noteworthy examples.

Second Life is an online virtual world, developed and owned by the San Francisco-based firm *Linden Lab* and launched on June 23, 2003. Reporting up to 1 million of regular users in 2013 and around 800 000 active user accounts in 2017 [Lab13], Second Life enjoyed great popularity by the end of the decade of 2000s and beginning of the 2010s. While sharing many characteristics with the MMORPG genre, their creators have always emphasised that Second Life is not a game, not having a set objective.

This virtual world comprised the technology to articulate it, such as client, messaging protocols and so, as well as the server infrastructure. Thus, Second Life only exists as long as their creators keep the service alive. Despite being a service provided by a third-party entity, the social nature and in-game degree of freedom offered by Second Life, together with its popularity among wider audiences, made it a popular research tool in several fields. Second Life proved to be a interesting study case for technical NVE-related topics such as avatar traffic analysis and mobility ([KC08], [Lia+09]), dynamic balancing of virtual worlds ([LB10]) or traffic modelling ([Fer+07]). Health-related applications were also noteworthy as shown by studies on health-related activities ([Bea+09]) or the potential of virtual worlds in medical education ([BHW07]). But where Second Life has been especially popular, like happened with other NVE systems, is in the context of e-learning, with multiple studies dealing with its potential applications in the field, such as [Hay06], [GHT08], [WI09] or [Get+10].

It is noteworthy though the popularisation in recent years, both among the gaming and research communities of the *sandbox* games. A *sandbox game* is a “style of game in which minimal character limitations are placed on the gamer, allowing the gamer to roam and change a virtual world at will. In contrast to a progression-style game, a sandbox game emphasizes roaming and allows a gamer to select tasks. Instead of featuring segmented areas or numbered levels, a sandbox game usually occurs in a “world” to which the gamer has full access from start to finish” [Tec]. While the terms *sandbox* and *open world* are sometimes used interchangeably, they refer to different concepts and they are not synonymous. It can be argued that the first example of sandbox-like game was the text adventure *Colossal Cave Adventure* [RB06] (1976) which allowed free-roaming exploration. In the 2000s several titles such as *Dwarf Fortress* [Rab15] (2006, by Tarn Adams) or *Infiniminer* [NSR16] (2009, by *Zachtronics LLC*) further popularized game dynamics built around the sandbox concept and also influencing *Minecraft* (2009, by *Mojang*) a landmark in the massive popularization of the genre.

Minecraft does not only offer a sandbox experience, but incorporates multiplayer features which fully qualify it as a potential NVE system. Minecraft provided freedom to explore and the ability for the player to modify at will vast procedurally generated virtual world, but also the possibility to extend the game with all kind of community mods and expansions, as well as running private servers that does not depend on any third-party service to operate. Thus, Minecraft not only gained the favour of the general public but also catch the attention of researchers that saw the title as a suitable canvas for different research areas [ES14] [NSR16]. The game has been used in the field of network traffic optimisation where [VDK13] addressed the scalability of the game servers, while [Als+15a] and [CHK17] used *bots* for performance evaluation. E-learning is a field where Minecraft has proven to be especially popular with multiple papers dealing with artificial intelligence (AI) learning courses [Bay12], developer collaboration platforms [BB13], computational thinking [Rep+14] or Computer Science teaching [Ach+16].

This trend where commercial NVE systems are used as platform for further research developments can be explained by the high cost of creating from scratch a research NVE platform, both in terms of resources as the multidisciplinary profiles required to provide a satisfactory user experience according to the available technology and current standards in usability, immersion and visual language.

1.2 Motivation and goals

The motivation of the present work is to characterise, analyse, model and simulate NVE network traffic roots in the omnipresence and importance of these kind of systems in today's world, ranging from entertainment applications to *pervasive games / serious games* related to activities such as e-learning, e-training, health-related activities. These modern scenarios already provide by themselves challenging technical questions related to the scalability of the technology itself and the interaction with humans. But this is also a fast-moving scenario where new technologies are already reshaping some of the paradigms, such as the irruption of cloud computing [Cho+12], and others are getting close to make a breakthrough to the mass consumer market, such as VR [Gun+17]. Thus, the networking aspects of NVE systems keep on posing research questions in a world with a level of interconnectivity and widespread of personal computing power without precedents.

The evaluation of network performance is the obvious field that directly benefits from studies on NVE network traffic, with applications ranging from optimisation of network equipment [Bor00] to properly sizing network infrastructures [Fen+02], [Bei+04]. A deep understanding of the internal dynamics of the network traffic has direct applications in the study of network QoS, defined as the description or measurement of the overall performance of a service, particularly the performance experienced by the users of the network. The study of the QoS involves the quantitative measurement of network parameters such as packet loss, bit rate, throughput, transmission delay or jitter. In this scenario accurate network models focused on NVE are a valuable tool for the study of such network parameters and the overall QoS for services that constitute a multi-billion business, as in the case of video games.

The QoS has also an impact on the perceived QoE, which can be defined as "the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state" [Bru+13]. The success and acceptance of applications based on NVE systems depends largely on providing a satisfying user experience [MSB13] and while meeting proper levels of network QoS is not sufficient condition to guarantee a good QoE, it is a necessary one. Thus, NVE network models would have a direct application in the study of QoE through QoS [Fin13].

The above motivation and goals are the result of the work performed during the period in which the author was a member of the Department of Computer Technology and Architecture of the University of Seville as a research fellow (2009–2013) funded by the program *Incentivos para la formación de personal docente e investigador predoctoral en áreas de conocimiento deficitarias* sponsored by *Consejería de Educación* of *Junta de Andalucía*, Spain. Thus, thread of the present work can be traced back to the papers published by the author on the subject. This work can be divided into the efforts put into modelling and those into simulation. Moreover, the later professional experience of the author has shaped some of the lines proposed as future work, such as the distributed execution of simulations.

Regarding the modelling aspects presented in this work, they go back to [Fon+11a], a paper where initially arose the need for a detailed study of the network traffic generated by a NVE system used by projects related to e-Health and e-learning. The preliminary network traffic study performed in this paper was limited to packet rate and bandwidth and aimed to provide tools to size properly the network infrastructure, keeping it from negatively impact the QoE of the users. The work in [Bor00] about *microscale* traffic modelling inspired the idea of creating more detailed models for NVE traffic as a more powerful tool to address the goals of the first paper. Thus, the journal article [Fon+12b] addressed the microscale modelling for the NVE software OWL, providing models for both the synchronisation of the shared virtual world and the audio transmission, as well as an study on the statistical discrepancy between the proposed models and the experimental network traffic that helped to define them.

Once the modelling process had reached a certain degree of maturity, the efforts shifted to designing and implementing simulations around the NVE traffic models. The initial challenge was choosing the proper simulation framework, which motivated the publication of [Fon+10], a comparison between *ns-2* and *ns-3*, two of the more popular network simulation frameworks at the moment, based on software metrics to assess the quality and potential technical benefits of each tool. This paper sparked interest in the simulation and performance evaluation community, reflected in the number of citations (33 by 2019) and in obtaining the *Best Paper Award* in the "13th *Communications & Networking Simulation Symposium*". The follow-up article, [Fon+11b], went into more details in the technical comparison between *ns-2* and *ns-3* by providing further code metrics for assessing code quality. From these two studies derived the technical decision to use *ns-3* framework as the foundation for later network simulations.

Based on the proposed NVE models and the *ns-3* framework, the study in [Fon+12a] put them to work together implementing simulations and comparing the generated traffic with the experimental one. This paper also started to inquire in the resource profiling of NVE servers by collecting performance metrics under different workloads. Finally, [FCS13] increased the amount of experimental data for modelling purposes by defining a more sophisticated NVE tested that relied on automation for better scalability, and improved the statistical methods for the modelling process. This paper was the result of a research stay at the Department of Information Engineering at the University of Pisa, Italy.

Last, the refinement of the statistical methods and automation of analysis procedures presented in this work are derived from the author's professional experience developed during his period working with distributed infrastructures for supercomputing ([Sha+14]) at the *e-Science Group* of the *Academisch Medisch Centrum* (2013–2016), and as system administrator and *devops* at *SURFsara* (2013–present), providing supercomputing and grid computing support to experiments such as *ATLAS* [CERb], *ALICE* [CERa], *TROPOMI* [Age], or *XENON* [Proh] among others. This recent experience has motivated an important part of the *Future work* proposed in the present work and the Appendix A, where it is briefly discussed the suitability of high-throughput computing (HTC) tools and techniques to exploit the parallelisation of NVE simulations based on *ns-3*.

Based on the above motivation and work history, the specific goals pursued by the present work are the following:

- Characterisation of NVE systems, providing an accurate definition of their characteristics and elements, from both the technical perspective as well as how the virtual world and users are modeled withing the virtual world.

- Choosing a software system that is compliant with the provided characterisation of NVE systems. This choice will be used as study case for the analysis and modelling procedures that are proposed in the present work.

- Definition and implementation of experimental testbeds where the chosen study case will be deployed and used to generate network traffic in a controlled environment. Moreover, guidelines will be defined to determine how to perform the data generation and collecting in the testbeds.

- Definition of an analysis and modelling pipeline for NVE network traffic. This pipeline will be based on previous literature and studies on network traffic for video games which are compliant with the characterisation for NVE systems proposed earlier. Thus, it will be a refined of existing techniques compliant with the specific characteristics of the NVE systems.

- Application of the analysis and modelling pipeline to the network traffic generated in the testbeds by the chosen study case. The result of this goal will be network traffic models that will accurately describe the behaviour of the NVE network traffic.

- Implementation of simulations based on the models for the chosen case of study generated by the analysis pipeline for NVE network traffic. The simulation results will help to assess the validity of the network models.

The above goals are addressed through each one of its chapters as follows:

- Chapter 1 provides an overview of the relevance of the NVE systems and its presence in the video game genre, as well as the motivation and goals of the present work.

- Chapter 2 provides a forma characterisation for NVE, presents OWL as study case for NVE systems to test and validate against it the processes and models proposed in this work, and describes the testbeds based on OWL where experimental network traffic is generated in a controlled environment. Appendix B contains further details about the tools and automation techniques used to build the testbeds.

- Chapter 3 reviews the literature about network traffic modelling of systems, many of them video games, that qualify as NVE according to the characterisation in Chapter 2. From this study is derived an *analysis pipeline* aimed to define microscale models for NVE network traffic, as well as assessing their accuracy by measuring statistical discrepancy. Appendix C contains the implementation in *R* of the proposed network models as well as other tools for the statistical analysis, such as the statistical discrepancy metric $\hat{\lambda}^2$.

- Chapter 4 applies the analysis pipeline proposed in Chapter 3 to the experimental network traffic generated by OWL in the testbeds defined in Chapter 2. The results of this process are several models to describe the behaviour of the network traffic in charge of synchronising the OWL virtual world across all the players engaged in the sessions. Appendix D contains further analysis details in the form of plots and data tables left out of the chapter due to space constraints.

- Chapter 5 designs and implements simulations based on the models proposed in Chapter 4 and the *ns-3* network simulation framework. The results from these simulations are used to validate the models.

- Chapter 6 enumerates the conclusions for the present work and the potential lines of future work. Appendix A further expands the line of future work focused on parallelisation of the simulations from Chapter 5 by using distributed infrastructure and technologies from the field of HTC.

1.3 Summary

The present chapter provides a brief description of the concept of NVE and an historical overview focused on the technological landmarks that constitute its building blocks and the fields where it has an special relevance. Thus, the popularisation of the personal computers, the definition of new visual language paradigms for computer software or the Internet explosion are addressed. Regarding the fields where NVE are relevant, the chapter reviews its origins as a tool for military training, its popularisation as foundation to build video games in what has become a multi-billion industry, or the academic interest for its potential as research tool.

The chapter describes de motivation for the present work based on the above importance of NVE systems and specially the importance of understanding the network traffic that articulates them. Moreover, it provides an outline of the timeline of the work that supports the present doctoral dissertation, including references to the papers and articles derived from it. Finally, the chapter enumerated the goals for the present work, derived from the above motivation and shaped by the work performed.

2 Networked virtual environments: characterisation and case study

David: Is this a game or is it real?
Joshua/WOPR: What's the difference?

WarGames, 1983

The present chapter aims to provide a detailed definition of the concept of networked virtual environment (NVE), this characterisation will prove helpful to identify software systems that, sharing a simulation over a data network, can be considered as NVE. Attending to this definition, there is a type of software where many of its exponents fall into the category of NVE, and this is the video game one, where many of the genres and titles are built around the concept of virtual world and the representation of the player in the form of an *avatar*. Genres such as first-person shooter (FPS), role-playing game (RPG) and their online version, the massively multiplayer online role-playing game (MMORPG), action, sports or racing games are all examples of this category. Real-time strategy (RTS) games would not mostly qualify regarding the player takes control of whole battalion of units, but any of the unit representations specifically represents the player within the virtual world. Section 1.1.2 provides an historical review detailing the convergence of video games towards the NVE paradigm. Adding part of the video game genre to the NVE category is not only an important addition in terms of weight in popular culture and entertainment industry, but also brings with it an extensive literature focused on networked video games that can be reinterpreted and analysed from the perspective of the NVE.

Having a clear characterisation to evaluate if a given software system can be considered or not as a NVE, the present chapter presents Open Wonderland (OWL), a software that matches these NVE requirements and will be helpful to put into practice the analysis strategies and techniques that will presented later in the current work. The last part of the chapter describes the experimental OWL testbeds built to capture network traffic to be used to generate statistical models describing its behaviour. These models will allow the later implementation of simulation tools and its use in the different fields that motivate the study of NVE network traffic, such as performance evaluation, network infrastructure sizing or quality of experience (QoE) derived from network quality of service (QoS).

2.1 Networked virtual environment

As stated in Chapter 1, a NVE can be broadly defined as a software system providing a simulation shared over a network where users located in different geographical places can interact among them in real-time. The existing literature dealing with NVEs provides a sparse set of characteristics that are common to this kind of systems [Tre03]. These characteristics can be broken down into three main categories. First, the *NVE software*, which implements the logic of the NVE system, it defines, hosts, updates and synchronises the simulation shared among the users. Second, the *NVE hardware*, which constitutes the physical layer, runs the NVE software and provides the input and output devices that allow users to interact in real-time with the shared simulation and also the network hardware required to access the communication channels. And third, the *data network*, which provides a communication channel that interconnects the hardware devices where the NVE software is run. These categories are described in further detail below.

2.1.1 NVE software

The NVE software orchestrates the whole simulation share among the NVE users. It keeps a representation of this simulation, its entities, properties and states. It also updates the virtual world with the input provided by users while trying to keep a consistent and synchronised view of the virtual world for all the users participating in it. Most modern NVE software are applications that run on top of a complete software stack comprising an operating system, hardware drivers and network protocol stacks.

The software architecture for NVE software can greatly vary, using different techniques to host the virtual world, provide persistence and synchronize and propagate state changes of its entities. Regarding the software structure, most of the software for NVEs can be classified as client-server [SZ99] or peer-to-peer [BAS13], [YK13]. The first rely on a central machine running the server software which hosts the virtual world, gather the updates and synchronises the views of all the NVE clients. The peer-to-peer ones host the shared simulation in a collective and non-centralised way, every peer acting as client and also server and using different synchronisation approaches to guarantee coherence.

Below is an enumeration of the defining elements and characteristics that can be found in NVE software.

Virtual World: is a computer simulation shared by users over a data network. This simulation is the representation of a reality following a set of rules that define the possible interactions of the entities contained within the simulation and how their states can evolve regarding these interactions. The NVE software implements the logic that defines the virtual world, from basic physics rules that govern the virtual world to the mechanics that shape the interaction, goals and rewards that users can experience within the virtual world.

Avatar: is a graphical representation used by a person to identify himself or herself in the context of a communication network or virtual community, providing the sense of presence [Çap99]. In the context of the NVEs, the avatar represents the user inside the virtual world. This entity will be controlled by the user and it will be its proxy between him/her and the rest of the entities within the virtual world. In the avatar model, the user interacts with the game through a single representative character and the player actions are defined in terms of commanding the character. The avatar exists at a particular location in the virtual world and can influence only the immediate locality. Software built around this avatar interaction model typically have either a first person perspective, where the player looks through the eyes of the avatar, or a third person perspective where the player follows an avatar in the virtual world. FPS games, RPG, action games, sports games and racing games are all examples of game genres that have an Avatar interaction model.

Real-time interaction between the users: but also between users and virtual world entities. An action on the user side involves an immediate interaction and result within the simulation that can be perceived by the user him/herself and the rest of participants. We must differentiate the concept of "real-time" depending on the context. From the user perspective, real-time refers to the reaction to events at the same time as they unfold, thus a change in the virtual world as consequence of the user input . From a computing perspective, real-time refers to performing actions within the system matching certain operational deadlines. Events, i.e. user input, have to be processed and the virtual world updated within defined time boundaries by the NVE system to minimise latency between user actions and system reaction, providing final users with the sensation of instant responsiveness [RG14], [Fin13], [Bru+13].

Communication channels: the simulation is shared among users over a data network where users not only can see each other's representation (or *avatar*) within the game, but they can also communicate beyond visual ways. The most common communication channels are written text in the form of chat rooms or instant messaging, and audio in the form of public conversations or private voice calls.

2.1.2 NVE hardware

The NVE user will require a computing system to access the virtual world, a device with a certain degree of computing capabilities to be able to run the required software to join the NVE virtual word, interact with it and receive a representation of it. This device acts as client of the NVE system and provides the link between the human user and the rest of the NVE. The most common option is the usage of microprocessor-based architectures equipped with RAM memory and storage memory for persistence. Moreover, due to graphical requirements it may be necessary to include dedicated hardware such as graphics processing units (GPUs).

Personal computers have been the traditional client devices for NVE systems. Both desktop workstation and laptop actor provide all the hardware requirements to run the NVE client software, being usually equipped with input devices such as keyboard and mouse, as well as connectivity to different network technologies. Later years have witnessed the emergence of handheld devices such as tablets and smartphones that can be also suitable NVE clients due to their computing capabilities or as input accessories, bringing new input methods into scene such as touch screens, a range of built-in sensors such as accelerometers and gyroscopes [Geb+14].

Below is an enumeration of the defining elements that can be found in

Control and input devices: the computing system has to count with input devices to allow the user to interact with the virtual world and control his/her avatar within it. These devices will translate the user actions into virtual world events. The most common user interactions involve avatar control and communication between NVE users using text or voice. Avatar control can be performed using keyboards, gamepads or pointing devices such as mouses or trackballs. Keyboards also allow to perform text input while microphones together with audio hardware allow audio input. Touchscreens are the norm in modern tablets and smartphones. Dedicated keyboards and pointing devices are usually replaced in these devices with controls directly displayed and operated from the tactile screen. Moreover, the new batch of handheld devices open new input possibilities thanks to the incorporation of built-in sensors. Moreover, later years have witnessed an irruption of the Internet of things (IoT) [WCJ14] or virtual reality (VR) technology [Gun+17], [He+17] to broader audiences, the later aimed to take immersive virtual world experiences to a new level.

Below is an enumeration of the defining elements and characteristics that can be found in the hardware supporting a NVE system.

Output devices: every user has a representation within the virtual world hosted by the NVE system. This virtual world stores all the entities within the simulation and all the details about location, properties and state [XS16], [Dai+09]. However, users are usually provided with just a partial representation of the virtual one, capturing the simulation area that can be encompassed from his/her location within the virtual world and following a set of physical rules of the simulation, like the field of vision. The most common trend is providing the user with a 3D render image of the virtual world from his/her perspective, this may require the presence of dedicated hardware such as GPUs. Other possible virtual world representations such as 2D or even ASCII based can be found on niche NVE, responding a technical limitations (underpowered graphic hardware by the time of their implementation) or merely aesthetic considerations (like retro game revival trends). Audio is provided as an extra to enrich the immersive experience within the virtual world, a computing device will require specialized audio hardware to decode and play sounds part of the virtual world environment, as well as peripherals such as speakers or headphones.

Data network: interconnects the hardware components of the NVE system, allowing the NVE software to share the virtual world among users. There are numerous physical channels that can be used to implement modern data networks, from wired technology such as Ethernet, to wireless networks based on WiMAX, Wi-Fi or G3/G4/G5 technology. The geographical extension of the network can also range from Local Area Networks confined to a room to world-wide ones, being the Internet the maximum exponent. On top of these technologies there is a network protocol stack where TCP/IP is the current de-facto standard. The network communications of a NVE system can be structured following a classic client-server architecture or peer-to-peer (P2P) approach [HL04] [SJT06] [BAS13].

2.2 Open Wonderland

Open Wonderland is a Java open-source toolkit for creating collaborative 3D virtual worlds, originally conceived as a tool for collaborative working by Sun employees [Yan+04]. It has been chosen in the context of the present work as a representative example of NVE and it will be used in later chapters as a case study to test the proposed analysis and modelling procedures defined for NVE network traffic.

According to the characterisation of NVE proposed in Section 2.1, OWL meets all the requirements to be considered as a representative of the NVE systems: it is based on *software* components running on *computer hardware* and provides a *virtual world* shared over a *data network* where users are represented by *avatars* that can engage in *real-time* interaction between them and also with other elements of the virtual world.

In addition of being an example of NVE software, OWL is also an interesting case study due to its expandability and suitability as a tool in research fields related to the motivations of the present work, enumerated in Section 1.2. For example, OWL has been used as foundation to build *persuasive systems*, focused on motivating healthy lifestyle habits [Fog02], such as *Virtual Valley* [Rom+10], [Cas+10]. OWL has been also used for e-learning studies such as [Gar+11]. Other studies such as [FSC12], [Fon+12a] or [FCS13] have used OWL as a case study for performance evaluation of NVEs, focusing on network and server resources respectively.

Thus, choosing OWL as case study for the present work is twofold: on the one hand due to its NVE condition it will help to prove the validity of the analysis and modelling process that will be proposed in the present work. On the other hand, if meaningful network traffic models are generated as result of this analysis process, they can prove useful in any of the fields that motivate the study and where OWL is a relevant tool, further confirming the interest in the study of NVE network traffic.

OWL has several other technical characteristics that make it especially suitable for its study and the design of experiments based on it. These characteristics mostly derive from its *open source* nature and design choices, making it an open platform for new developments [KY11], allowing the publication and distribution of derived works and giving organisations and individuals the possibility to deploy the system on their own infrastructures without having to rely on third-party services, which is especially crucial in applications that may involve highly confidential data, such as medical records [Gar+09].

2.2.1 Architecture and internal components

The OWL version used in the present work is *Project Wonderland 0.5*, its architecture is shown in Figure 2.1 [Proc]. OWL is subdivided in several components and third-party projects that work together to deliver a NVE experience. These elements are enumerated and detailed below.

Wonderland: comprises both the core of OWL client and server as well as a set of modules that provide key functionalities such as security, shared applications, avatars and so on. It also contains the web administration server. Specifically, the *shared* application feature allows sharing applications among different users. Some of these

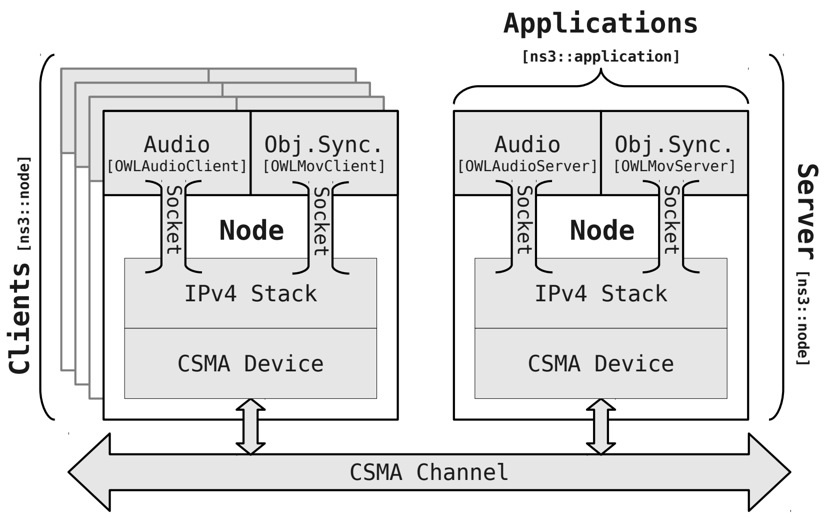


Fig. 2.1.: Open Wonderland client-server architecture

applications are already integrated in Wonderland, like the multi-user PDF viewer and the *SVG White board*. Users can also share additional external applications installed in the server (like Firefox or OpenOffice) using the *Shared Applications Server*.

Wonderland Modules: a repository for OWL extension modules which expand its functionalities. Some of these modules are shipped by default within the OWL binary releases. There are also experimental modules provides by both project developers and community members.

MTGame Graphics Engine: a high-performance graphics engine that extends jMonkeyEngine. MTGame adds multi-threading capabilities for improved graphics performance [Foud].

jVoiceBridge: a pure-java audio mixing platform providing real-time immersive audio (via VoIP) with time distance attenuation and a selectable range of qualities as well as a companion software phone, called *softphone*' that allows phone calls between users within the virtual world. It supports mixing high-fidelity, stereo audio at up to CD quality [Fouc].

Moreover, OWL includes several open source projects as internal components, such as graphic or audio engines, applications servers and so on. Some of these third-party components are listed below.

Darkstar: a Java platform created by Sun Microsystems for scalable communications and persistence in games. OWL includes a Darkstar service that manages all client and world state. Nowadays the project development is officially halted but a community fork has been created, called RedDwark Server [Proe].

Glassfish: is highly scalable, open source pure-Java application server that provides several functionalities for the Java EE platform such as RMI, XML and web server. OWL is based on an embedded instance of the Glassfish server. Wonderland web applications include web-based management of the server and worlds, a content repository for hosting all world data, and an integrated single-sign on system used to maintain identity across Wonderland services [Cor].

jMonkeyEngine: a 3D game engine written entirely in Java. It provides core graphics APIs, including graphics primitive and shader support. The OWL graphics system is based on these core APIs, with some extensions from MTGame to support multithreading [Teab].

2.2.2 In-game elements and interactions

Within the OWL virtual world, the user is graphically represented by a 3D object known as *avatar*, the virtual projection of the user perceived by other players within the same gaming session. Its movements and behaviour are determined by the user behind it, using his/her keyboard, mouse or any other computer input peripherals.

The virtual world can contain all kind of objects, which can be 3D objects like pieces of furniture, buildings, etc. or 2D objects like screens with embedded applications (web browsers, word processors, and so on). The usual way to model the spatial relationships between objects is using a scene graph. Each object is a node (or *cell* in OWL terminology) in this graph. The *cells*, representing any volume of space of the virtual world, are organized in a graph with a tree hierarchy [Prod]. Figure 2.2 shows an example of the *cell tree*.

The in-game interactions within OWL can be divided into three categories: avatar movement, audio communication and interaction with embedded apps.

Avatar movement and object interaction: these interactions are related to the location of the avatar within the game scenario and the other avatars and objects in it. The result of these interactions are the update and synchronization of the shared simulation between all the game users and their graphic representation.

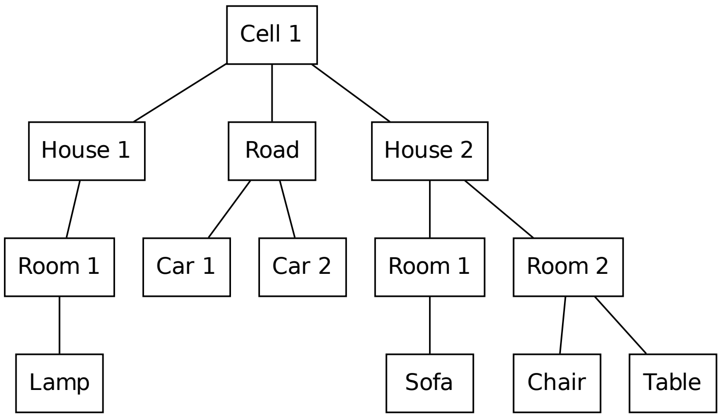


Fig. 2.2.: Structure of a Open Wonderland cell

Audio communication: involves the transmission of audio between users in the form of private conversations or audio being broadcast to all the users within the listening range.

Embedded applications: these are applications within the game, such as browsers, digital boards or PDF readers. These kind of interactions are very heterogeneous since they are strongly linked to the nature of the application. Their study and modelling are out of the scope of the present work.

Each one of the above type of interactions generate a characteristic and distinctive type of network traffic. Each type of interaction has optimized network flows designed to better suit their requirements: reliability, data integrity, real-time and so on. These network flows are further discussed in section 2.2.3.

2.2.3 Networking

OWL is based on a *client-server* architecture [Fouf]. The OWL server must have an IP address known and reachable by the clients. Initially, a client is disconnected from the server until it calls a *WonderlandSession.login()*. If this succeeds, the session goes into the *CONNECTED* state. A client may connect multiple sessions to the same server. Once a session is connected to a server, connections may be added to it. Each connection in Wonderland has a unique type for sending different types of data. For example, a client may use one connection for sending *cell* data, and another one for sending voice communications data. Clients may use as many connections as they need for their interaction with the server. The only limitation is that a client may only have a single connection of a given type connected to a given session. Clients may also use multiple sessions to get multiple copies of a single connection type.

Once in the *CONNECTED* state, messages can be passed from a client to the server, from the server to one client (not necessarily in response to a client message or request) and also the same message can be sent by the server to a number of clients. From version 0.5 on, there is no more client-client communication, that is, all messages from clients go directly to the server. The traffic between clients and server/s can be broadly divided into *object synchronisation*, *audio*, and *application data* [She+03]. The *object synchronisation* traffic uses TCP protocol to ensure the virtual world updates are received by all the clients engaged in the same session, this type of traffic will be subject of further analysis in Chapter 4. The *audio* traffic uses UDP for the propagation of voice and audio content. While UDP does not guarantee the delivery of the packets, it is acceptable a certain degree of packet loss without compromising and spoiling the user experience. The studio of the audio traffic has been left out of the scope of the present work. Finally, the *application data* traffic is highly dependant of the nature of the embedded applications that generate it and it has also been left out of the scope of the present work.

2.3 Experimental testbeds based on Open Wonderland

Using OWL as case study for NVE network traffic modelling requires to have available experimental network traces generated and captured under controlled circumstances. To this end, two testbeds based on OWL have been defined, the first one, labeled as Open Wonderland TestBed 1 (OWTB1), follows one of the classic approaches on literature where human players volunteer to participate in gaming sessions which network traffic is captured an later analysed.The second testbed, labeled as Open Wonderland TestBed 2 (OWTB2), is an attempt to be more scalable by using basic bots to control the avatar, replacing human operators and the constrains associated to the availability of volunteers. Nonetheless, both testbeds are *client-server* installations where clients connect to a standalone OWL server. Both testbeds are described in further detail below in the next sections.

2.3.1 Open Wonderland Testbed 1

Open Wonderland Testbed 1 (hereafter OWTB1) was the first testbed to be deployed in the laboratory of the Department of Computer Technology and Architecture of the University of Seville. This testbed was composed by heterogeneous hardware and operated by a group of volunteers. This testbed was aimed to capture initial OWL network traffic to help in the first evaluations and the initial definition of the analysis and modelling processes. Therefore, this is a heterogeneous platform in both hard and soft terms. The structure and topology of OWTB1 is shown in Figure 2.3.

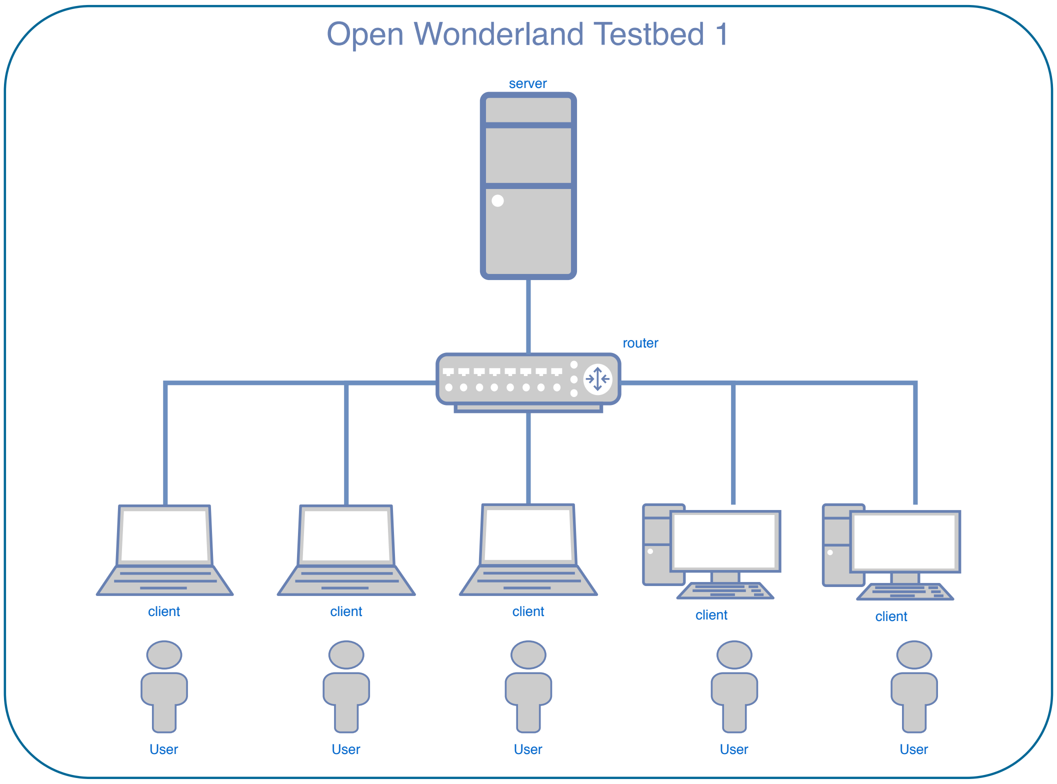


Fig. 2.3.: Structure of Open Wonderlad Testbed 1

Hardware: OWTB1 was implemented using a set of PC machines from different generations and specifications. The first testing session followed the bring your own computer (BYOC) principle [RHS10], where all the volunteers brought their own machines and connected to the Ethernet infrastructure and OWL server provided for the experiment. This situation does not differ greatly from any other regular OWL deployments observed *in the wild* which deal with different hardware settings. Despite the differences in computing and graphic resources, all the machines met the minimum requirements to run the OWL client software [FSC12]. As for the server hardware, it had enough resources to not constraint the performance of the system under the planned workloads. Table 2.1 shows a summary of the hardware settings of the OWTB1 computers.

Network: the OWTB1 nodes were interconnected using Ethernet technology. The network configuration consisted in a single Ethernet 10/100Mbps switch, provided by a Linksys WRT54G v7.0 with stock firmware. The switch interconnected the server and clients in a single collision domain. The usage of wired technology and ISO Layer 2 device aimed to provide an isolated network scenario that minimised

Tab. 2.1.: Hardware setup for OWTB1

Role	Processor	RAM	GPU
Server	AMD Phenom x4 2.6 GHz	4GB DDR2	GForce 8200
Client 1	Intel Dual Core x2 3 GHz	2GB DDR2	Radeon X300
Client 2	Intel C2D x2 2.53 GHz	3GB DDR2	GMA 4500 MHD
Client 3	Intel C2D x2 2.5 GHz	4GB DDR2	GeForce 8600 GT
Client 4	Pentium IV HT 3.2 GHz	512MB DDR	Radeon 9700
Client 5	Intel C2Q x4 2.4 GHz	6GB DDR3	GForce 9700

network delay. The latency induced by the Ethernet 100Mbps network was an order of magnitude smaller than the times between consecutive packets generated by the OWL clients [Yan12] [Aza05]. Moreover, the 100Mbps bandwidth was more than enough to support all the traffic generated by the OWL instances. Studies such as [NC04] and [Ami+13] follow a similar approach, were the different clients connects to the same Ethernet segment.

Software: OWL is a project based on Java technologies, it runs over a Java Runtime Environment (JRE) and therefore it is agnostic to some extend to the underlying software stack (operating system, drivers, etc.). OWTB1 used two different software configurations for server and clients respectively. GNU/Linux was chosen as operating system for the server machine, specifically Debian 6.0 (updated with software packages of the Debian testing branch in June 30^{th} 2011) optimised for amd64 architecture (x86_64, 64-bit extension of x86, 32-bit). GNU/Linux is widely used in servers, where it runs all kind of applications, including Java-based ones, due to its reliability, availability, configuration possibilities and relatively lean resource requirements.

Although the JRE is enough to run the OWL client, all the client machines were equipped with the Oracle Java Development Kit (JDK) for the corresponding 32 or 64-bit architecture. The JDK installed on the server was the one provided by the now-defunct Sun Microsystem, the Sun JDK v1.6.0, compiled for 64-bit architecture. OWL v0.5 nightly build (corresponding to June 30^{th} 2011) was the server version chosen for OWTB1.

Machines running the OWL clients were powered by Microsoft Windows XP and Windows 7. Windows XP is a 32-bit only architecture, while Windows 7 was present in OWTB1 with both 32 and 64-bit versions. The Windows-based systems were representative of the Microsoft-dominated domestic PC market [Sta12] at the moment of building the OWTB1. Despite the heterogeneous set of 32 and 64-bit software in OWTB1 (x86 vs x86_64), the network traffic should be mostly agnostic to this detail, being more influenced by other hardware and software details such as optimization, software architecture and so on. The server machine ran an instance of OWL v0.5 nightly build, corresponding to June 30^{th} 2011. Players used the client version shipped with the OWL server.

Gaming sessions: those performed on OWTB1 followed the guidelines defined in section 2.3.3 regarding session duration and player behaviour. The gaming sessions on OWTB1 were conducted by human players, following the trend of most authors that have based their studies on real human-driven gaming sessions [BA06] due to the complexity of using reliable models for simulating human player behaviour. In general terms, automating gaming sessions and removing the human factor would present a high degree of complexity. Using models to simulate human player behaviour and mobility is even more complex than using real players since it requires finding realistic models for player behaviour.

The gaming sessions on OWTB1 were performed by a total of 5 volunteers, students and members of the School of Informatics Engineering of the University of Seville. The volunteers received training on the usage of the OWL client and also basic guidelines about the in-game behaviour and rate of expected interaction between avatars. These guidelines are described in more detail in section 2.3.3.

All the testing sessions were performed in the same in-game scenario, a minimalistic virtual world where each user was within view and hearing range of the rest of the players. A total of 3 independent gaming sessions were performed. The number of concurrent users in each session was of 2, 3 and 5 respectively. The 2 and 3-client session showed the trend in the generated network traffic. Due to time and human resource constraints, the 4-client session was omitted, directly performing a 5-client session to confirm the trend observed in previous sessions.

2.3.2 Open Wonderland Testbed 2

Open Wonderland Testbed 2, hereafter OWTB2, was designed and deployed taking into account the experience acquired from its predecessor OWTB1. OWTB2 was deployed in one of the computer laboratories of the Department of Computer Technology and Architecture of the University of Seville. This scenario removed the BYOC factor present in the previous OWTB1, providing a hardware and software platform easy to configure and administrate. This, together with no further requiring volunteers, allowed to easily scale up the number of players in the experimental sessions performed on this new testbed. Thus, it was finally possible to increase the number of sessions in a homogeneous platform to better observe the impact of higher workloads on the network performance. Besides the above differences, OWTB2 is still based on the *client-server* architecture, following the same experiment guidelines that OWTB1 in terms of session duration, scenario and type of interaction between players. The structure and topology of OWTB2 is shown in Figure 2.4.

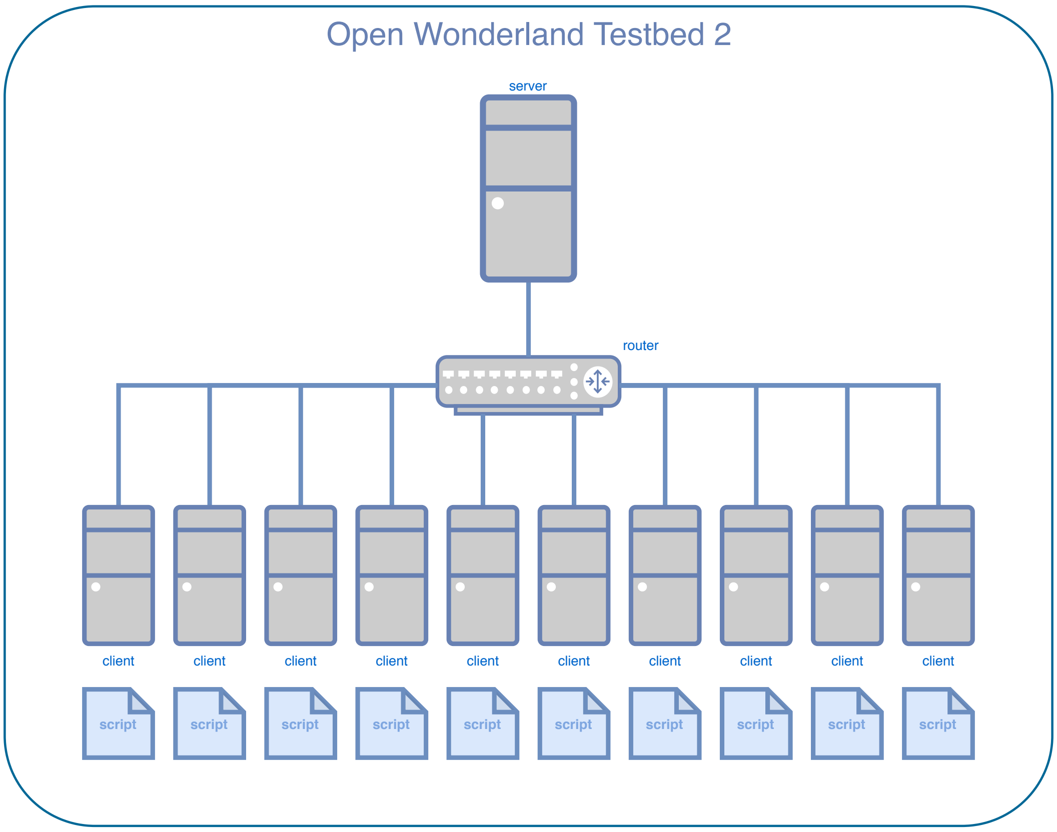


Fig. 2.4.: Structure of Open Wonderland Testbed 2

Hardware: the machines used in OWTB2 had identical hardware settings, details are shown in Table 2.2. This hardware supposed an improvement over OWTB1 in terms of computing resources and homogeneity of the testbed. One of the reasons to use identical computers was to remove the variability induced by heterogeneous hardware in OWTB1 to better observe the network behaviour of OWL. In OWTB2 both server and client nodes had identical hardware configuration for mere convenience reasons, regarding all the machines in the computer lab were identical. This settings exceeded the minimum software requirements to run OWL client. On the other hand, these settings allowed the server to support up to 10 concurrent clients engaged in a same gaming session.

Tab. 2.2.: Hardware setup for OWTB2

Role	Processor	RAM	GPU
Node	Intel i7-2600K 3.4GHz	8GB DDR3	GForce GTX 550Ti

Network: the nodes in OWTB2 were interconnected using the local area network available at the computer laboratory where it was deployed, a Gigabit Ethernet network providing a single collision domain. As in the previous testbed OWTB1, the values of the packet lag induced by the testbed network are one magnitude below those for network variables such as inter-arrival time (IAT) or inter-departure time (IDT), which are key in later analysis and modelling steps.

Software: OWTB2 kept the distinction between server and client software stack made in OWTB1. The server machine ran Ubuntu 12.04 compiled for i386 architecture and including all its respective security updates and a Linux kernel 3.2 optimised for i686 architecture.

The client machines shared a common software configuration running Microsoft Windows 7 64-bit with Service Pack 1 (SP1) installed. OWTB2 clients used the Oracle Java Runtime Environment (JRE) v.1.7, this JRE was already installed in the lab computers. Regarding Java implementations, they are retro compatible with previous specifications, there was no need to install a different JRE version. As in the case of hardware, software heterogeneity was removed in favour of an uniform testbed.

The OWL server in OWTB2 required the usage of a complete Java Development Kit (JDK), the chosen one was the OpenJDK implementation of Java v1.6.0 specification, compiled for i386 architecture. The usage of an Intel 64-bit hardware architecture (amd64) running both 32 and 64-bit software has a negligible impact over the network traffic results; the network communication in our study is not conditioned by the underlying CPU architecture. The server ran OWL v0.5 nightly-build from February 20^{th} 2012. This version did not provide any significant architecture or technical change from the one used in OWTB1 beyond minor bug fixes.

Gaming sessions: those performed on OWTB2 followed the guidelines defined in section 2.3.3 regarding session duration and player behaviour. Like in the previous OWTB1 testbed, all the testing sessions were performed in the same minimalistic game scenario. A total of 10 gaming sessions were performed ranging from 1 to 10 concurrent players.

Automation of player input in OWTB2

Bots are weak AI expert system software that can control a player during a gaming session, they are quite prominent in several genres such as FPS and MMORPGs [Wav01]. There are some examples where they have been used to study video game network traffic [Als+15b]. Following this approach, OWTB2 replaced human players with a sort of rudimentary *bots*, scripts which automatically injected input into the OWL clients to control the movement of the avatars. For this automation process player behaviour has been simplified, aiming to maximise the interaction rate and thus its associated network traffic. Although this simplification may seem not representative of the traffic obtained during real gaming sessions, the results from [Fon+12b] suggest that the impact of lower rates of activity in OWL sessions translates into greater IAT/IDT values and therefore more heavy-tailed distributions while keeping an analogous nature to those proposed in this study.

It could be argued that the more sophisticated *bots*, directly integrated within the OWL client, should have been used. This would have meant to bypass the hardware layer that, combined with the human intervention, determine the ratio of keystrokes and the related update events. It must be highlighted that the goal of this approach is not reproducing a human player behaviour, but providing an automated avatar generating as many keystrokes as a human player in front of a keyboard. While OWL provides several frameworks to create *bots*, there was no way to match the player's keyboard behaviour to get a similar rate of update events and associated network traffic.

Two different scripting applications were used to interact with the Windows API in charge of keyboard management. Each client machine had a AutoHotKey (AHK) instance [Foua], an open-source macro-creation and automation software utility. The AHK script generates cursor keystrokes, responsible of the avatar translation. To avoid any kind of correlation in their behaviour, the script determines the direction of the movement: forwards, backwards, turn right or turn left. This decision is made using a pseudo-random number generator following an uniform distribution [Foub]. The amount of time during which the avatar performs the selected movement is also determined using an uniform distribution. Independently of the erratic movement performed by each "automated client", they constantly generate updates about their respective position which are propagated all over the OWL system as TCP packets. More detailed information about AHK and its configuration can be found in the Appendix B, Section B.2.

A second scripting application, HotKeyNet [Teaa] , was employed to remotely control the functioning of the AHK scripts, allowing to launch and stop them from a single machine acting as if it were an operator. Further technical details about the automation scripts can be found in Appendix B, Section B.3.

2.3.3 Guidelines to conduct gaming session

The gaming sessions captured and studied on both OWTB1 and OWTB2 followed the same guidelines in terms of game duration and in-game player behaviour. Regarding the duration of the gaming session, each one was set at 11 min. The first minute of each session was considered as warm up, giving time to all the players to join the gaming session. This first minute has been discarded, leaving 10 min of useful network traffic traces for later study.

Studies such as [She+03] have classified the types of in-game actions and patterns to later focus on the network patters associated to them. One of the observed approaches has been the distinction between between *active* and *inactive* players, focusing on the first ones considered as more representative about the normalised gaming experience to be studied and modelled [Fen+02]. In [Fon+12b], it was determined that the inactivity periods of the so-called *inactive* clients translated into absence of traffic from client to server. The guideline for both testbeds was to maximise user interaction in order to reduce outliers and the tail effect in some network parameters such as IAT. A similar experimental approach where predefined behaviours have been chosen to maximize certain types of interactions can be found in [Bei+04].

2.3.4 Network traffic capture

The capture of the network traces in OWTB1 and OWTB2 has been performed following the same procedure. In both cases the chosen tool was the packet analyser *Wireshark* v.1.6.5 [Tea19b] and specifically *Tshark*, its terminal-based counterpart which provides the same functionality regardless the graphical user interface (GUI), facilitating the scripting and automation of the process. The command line tool *capinfos* was used to extract statistics from the `pcap` files generated by `tshark`, Bash scripting to articulate the analysis workflow and the statistics language R for calculations and plots. Captured network packets missed 4 bytes corresponding to the frame check sequence (FCS) for Ethernet datagrams. While Wireshark supports the capture of FCS [Prog], many network interface controller (NIC) drivers do not handle the FCS field to the operating system and therefore these bytes do not reach the analyser tools. This has been taken into account in the later study of the packet size.

Following the example of other studies about gaming traffic, the capture of the packets was made at the server-side ([Fen+02], [Fär02], [CLW03], [Fen+05], [Cla05]). Each capture contained 10 min of network traffic. The traffic captures were started after all the clients had logged into the gaming session and their views of the virtual world had already been loaded, thus avoiding interference with initial synchronisation process and their associated network patterns [CC10]. These packets are mostly associated to the initialisation processes and are only present during the initial stage of the gaming session where they transport a copy of the virtual world environment to each client, not being significant for the stationary scenario of the traffic. Due to the simplicity of the in-game interactions it was not required a significant *warm-up* period to reach stationary state. A 10 min trace capture is representative enough of the network traffic dynamics to be studied [Mac87].

2.4 Summary

The present chapter provides a characterisation of NVE systems, a tool to identify software systems that can be categorised as NVE. The immediate application of this characterisation is to identify NVE systems in areas such as video games, and therefore being able to reuse and reinterpret the existing literature focused on them from the perspective of NVE. These allows to identify a lot of material to synthesize the analysis and modelling processes for NVE network traffic proposed in the present work.

The characterisation of NVE systems describe them as software providing a simulation, or *virtual world*, shared over a data network that is powered by computing devices and peripherals. The representation of the users engaged in the virtual world experience is done by the usage of an *avatar*. These avatars interact with each other and with other elements within the virtual world in real-time basis.

OWL complies with the NVE characterisation proposed in this chapter. This, together with its use as research platform in fields related to the motivation of the present work, make OWL a good candidate to test and validate the analysis and modelling procedures for NVE platforms later developed in this work. The open source nature of OWL and its architecture and technical characteristics make it also a good alternative to design experiments and generate network traffic to be used for modelling purposes.

Having chosen OWL as a case study for NVE network traffic analysis and modelling, the chapter also describes the testbeds built around it, OWTB1 and OWTB2. While they have some differences, such as hardware setup or the way avatars are controlled in each one, they still share the same client-server topology over an Ethernet network, network traffic capture procedure and the same guidelines to conduct experiments, where certain types of user interactions are especially favoured due to its interest from the modelling point of view.

3 Analysis and modelling pipeline for NVE network traffic

But magicians have calculated that million-to-one chances crop up nine times out of ten

Terry Pratchett, Mort

This chapter provides an overview on previous work analysing and modelling network traffic generated by different types of video games, using them to extrapolate and propose a statistical analysis pipeline specifically devised to analyse and model network traffic generated by networked virtual environment (NVE)s. The reason to rely on video game-related literature is its abundance and the fact that many video games overlap with the NVE definition provided in Section 2.1. This chapter is an effort to identify and filter the statistical devices that can help to shape the statistical analysis of the network traffic generated by the NVE systems defined the present work. For convenience reasons, when the term *video game* is referred it is implicit that refers to those which share characteristics with the NVEs.

An analysis pipeline consists in a number of interconnected processing steps that receive a certain data input and perform an analysis to generate some sort of output. The pipeline proposed in this chapter is a collection of steps and statistical tools that can be automated partially automated using a programming language (i.e. R [Fouh] or Python [Foug]), and workflow technologies [Sha+14].

The modelling strategy implemented through this pipeline is the so called *microscale modelling* [GS10], which models the network traffic based on two random variables, inter-arrival time (IAT) and packet size (PS). The aim of the proposed *statistical analysis pipeline* is to describe the steps researchers can follow to determine a model for each one of the random variables (IAT and PS) that describe the traffic at microscale level. Some of the steps will require the researcher to make decisions, such as the most promising theoretical distribution to fit into the experimental network traffic. The pipeline also include steps to determine how optimal those decisions have been, i.e. evaluation of Goodness of Fit, allowing the researcher to reconsider and iterate again over the process.

3.1 Background

Network traffic models have been traditionally developed under the assumption that packets or flows arrive according to a memoryless Poisson process [Kle75]. This implies that observed network traffic exhibits only short-term autocorrelations where the autocorrelation function of the observations becomes negligible after a certain finite lag. Later, self-similar models of network traffic became widespread in literature [Lel+93] [FM94], being successfully applied to different real-world traffic traces, such as variable bitrate (VBR) video streams [Ber+95] and World-Wide-Web traffic [CB97]. These models could capture the long-term autocorrelations measured in local area network (LAN) and backbone traffic, however there is no concise relation that describes how self-similar traffic affects queueing at switches or end-user performance.[GB96]. There have been other approaches for the video game network modelling, such as the study in the frequency domain [Fen+05], or using time series such as Markov chains or autoregressive-moving-average (ARMA) models [CB07] [CBA07].

The present work follows the microscale approach for network traffic modelling, consisting in examining the inter-arrival dynamics between individual packets withing a gaming session. Microscale models have been used previously to model the packet inter-arrival periods of telnet [Pax94] [PF95] and FTP [Dan+01] traffic. Some of the earliest applications of the microscale modelling to the video game network traffic can be found in [BDA99] and [BD00], but it is the paper from Michael S. Borella, *"Source Models of Network Game Traffic"* [Bor00], the work that defined the basic guidelines for the microscale network modelling as well as the measurement of discrepancy of these models against the empirical network traffic. This work laid the groundwork for many later papers on this subject, are numerous the literature applying these guidelines to video games from first-person shooter (FPS) genre, such as Half-Life [HB01] [Lan+03], Quake3 [LHR02] [LBA04], Counter-Strike [Fär02] [Fär04], Halo2 [ZA05], Unreal Tournment 2004 [Hüb08]. Network traffic generated by massively multiplayer online role-playing game (MMORPG)s has been also covered in works such as ShenZhou Online [Che+06a]. The real-time strategy (RTS) genre has been also subject of study from the microscale network modelling with representatives such as StarCraft [DPV05].

3.2 Analysis pipeline for microscale modelling

The microscale modelling for video game network traffic popularised in [Bor00] considers traffic as a flow of packets of different sizes, each packet separated from the previous and next by certain amount of time. Thus, the magnitudes that focus attention for the microscale modelling are *IAT / inter-departure time (IDT)*, period of time separating consecutive incoming / outgoing network packets and *PS*, the number of bits composing the network packet. These magnitudes can be considered as random variables which behaviour can be described using stochastic models.

The basic steps for the analysis pipeline for microscale modelling of the random variables IAT and PS, are enumerated below. Determining the model for each variable requires to execute the pipeline for data input specifically filtered and adequated to model such variable. Each one of these steps are described in further detail in the following sections. Figure 3.1 describes a flow chart for the pipeline.

1. Data filtering: network packets useful to study IAT or PS will be identified and isolated during these step.
2. Data preview: initial graphical representations and analysis of the raw data will help to determine how later steps will be performed. Statistical elements such as the plot of the Empirical Cumulative Distribution can help with this step.
3. Autocorrelation: determining the presence of self-similarity (or lack of) allows to determine the right modelling approach for the raw data. Proving the randomness of the studied parameter allows to go on with the microscale analysis pipeline, otherwise, other approaches such as time-series may be more appropriated for traffic of this nature.
4. Data Modelling: one or more statistical models will be evaluated to determine the most promising to describe the studied network parameter. This step will be conditioned by the discrete or continuous nature of the values observed during the preview step.
5. Goodness of Fit: the fitness of the models proposed in the previous step will be determined against several goodness of fit (GoF) techniques such as Q-Q plots or numeric discrepancy metrics.

3.3 Data Filtering

The *Data Filtering* step aims to extract the relevant network traffic from the captured traces for later statistical modelling. In the case of IAT traffic, the filtering will select the arrival times of the packet traces in the capture which will allow to calculate the time between consecutive arrivals of similar packets. The filtering for the study of PS only require to focus on the packet size of the payload exchanged through the NVE network traffic.

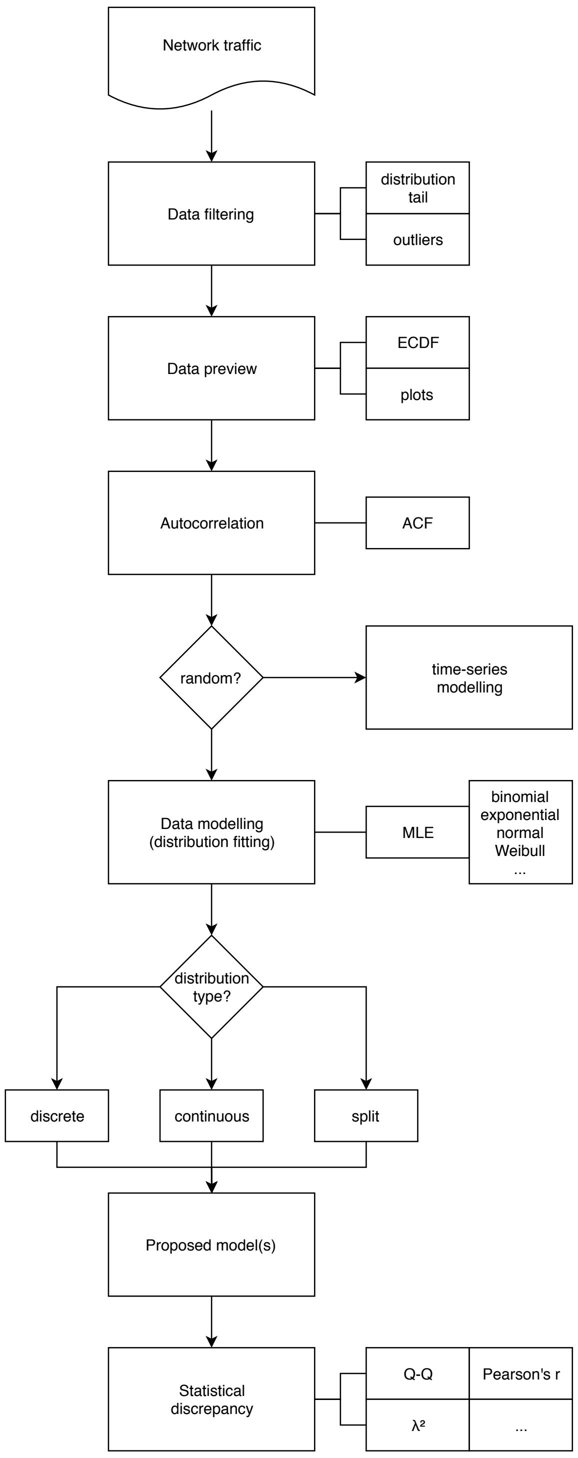


Fig. 3.1.: Workflow for the NVE analysis and modelling pipeline

There are both technical and statistical considerations to take into account during this step. Technical considerations are related to the nature of the traffic itself, allowing to determine which data flows are of interest for the later stages of the statistical analysis. The following list of considerations have been used to determine how to filter experimental in the context of the present world:

- network traffic source: client-generated traffic Vs server-generated one
- type of transport layer protocol: TCP vs UDP
- nature of the packet payload regarding the game stage: scenario load, user login, regular gameplay, etc.
- nature of the packet payload regarding the event it codifies: avatar sync, audio, app-related traffic, etc.
- evaluation of complete packet size Vs trace payload
- details related to the traffic captured tools used, i.e. lack of frame check sequence (FCS) field in traces captured by traffic analyser Wireshark.

The statistical considerations are mostly related to the statistical concept of *outlier*, that can be defined as *an observation point that is distant from other observations*. Outliers may be due to variability in the measurement or they may indicate experimental error. In case of the latter, the outlier can be removed from the data set [DGK75]. Moreover, some of the statistical tests may be overly sensitive to outliers (i.e. λ^2 [Bor00]). Another statistical consideration is related to the distribution tail and its impact on later steps of the analysis process [BD00] [Fär04] [BA06]. Later statistical analysis of experimental network traffic will include the criteria followed regarding outliers and their handling.

By the end of this step the raw experimental data will have been transformed in a data set containing the values that will allow to model the desired micromodelling parameters (IAT or PS) in the following steps of the pipeline.

3.4 Data preview and preliminary analysis

The *Data preview and preliminary analysis* step aims to provide a initial graphical preview of the experimental data used to model the random variable. Other statistical tools, such as applying basic distribution fitting can also be performed during this step to better identify potential models. This preview will help researchers to decide how to conduct later steps of the analysis pipeline.

The cumulative distribution function (CDF) [SW09] describes the probability that a random variable $\mathcal{X}$ with a given probability distribution will be found at a value less than or equal to x. For continuous distributions, it provides the area under the probability density function from $-\infty$ to x.

The empirical cumulative distribution function (ECDF) [Vaa00] is the cumulative distribution function associated with the empirical measure of the sample. This CDF is a step function that jumps up by $1/n$ at each of the n data points. The empirical distribution function estimates the true underlying CDF of the points in the sample and converges with probability 1 according to the Glivenko–Cantelli theorem [Tuc59].

While it is possible to base the measurement of discrepancy on the ECDF [Ste74], in this step it will be only used to provide hints whether a given data set can be described by a theoretical distribution based on the comparison between the data ECDF and the distribution CDF. While simple, this approach may prove to not suffice in situations where it is necessary to determine which distribution better describes the given data set.

By the end of this step, a graphical preview of the data set will be available, acting as input for later steps in the pipeline, specifically the *Selection of a probability distribution*.

3.5 Correlation and Autocorrelation

The *Correlation and Autocorrelation* step aims to evaluate the randomness of the data set thus determine if it is suitable for later microscale modelling based on probability distributions.

The term *correlation* is used in statistics and probability theory to describe the dependence between the values of two random variables [Haz95]. It is used to include a standardising factor ranging between -1 and 1. The formula for correlation is defined in Equation (3.1).

$$cor(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

$$cov(X,Y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N} \tag{3.2}$$

The *autocorrelation* concept is present both in the fields of signal analysis and statistics. For the fist one, it is the cross-correlation of a signal with itself, describing the similarity between observations separated between them by a certain amount of time. In statistics, the *autocorrelation* of a random process describes the correlation between values of the process at different times, as a function of the two times or of the time difference.

Let X be a repeatable process and i some point in time after the start of that process. i may be an integer if X is a discrete-time process or a real number for a continuous-time process. X_i is the value for a given run of the process at time i. It is assumed that the process X has defined values for mean μ_i and variance σ_i^2 for all the times i. For this scenario, the autocorrelation between times s and t is defined in Equation (3.3), where E is the expected value operator.

$$R(s,t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s} \tag{3.3}$$

If the function R is well-defined (there are no variance values equal to zero nor infinite), the autocorrelation value is in the range $[-1,1]$. A value of 1 indicates a perfect correlation, the series matches perfectly with itself after being shifted a determinate amount of time. On the other hand, an autocorrelation value of -1 means perfect anti-correlation or inverse correlation.

If X_t is a wide-sense stationary process then the mean μ and the variance σ^2 are time-independent, and further the autocovariance function depends only on the lag between t_1 t_2: the autocovariance depends only on the time-distance between the pair of values but not on their position in time. This further implies that the autocovariance and auto-correlation can be expressed as a function of the time-lag, and that this would be an even function of the lag $\tau = t_2 - t_1$. This gives the more familiar forms for the autocorrelation function (ACF) shown in Equation 3.4.

$$R_{XX}(\tau) = E[X_t \overline{X_{t+\tau}}] \tag{3.4}$$

The presence of autocorrelation in the studied network traffic can help to direct the modelling approach and the type of model to be used. While models based on time-series [CBA07] [CB07] rely on the presence of autocorrelation to determine if a Markov chain or ARMA model is suitable to describe the nature of the network traffic, works following a microscale approach such as [Bra06] or [DPV05] start measuring autocorrelation in the studied network parameters to determine the random nature of the traffic and the suitability of a statistical model derived from maximum likelihood estimation (MLE).

According to [Bra06], "an indication of whether or not a traffic exhibits Markov characteristics is the rate at which correlation between two samples decreases as the distance between the samples increases [Pax93]. For a Markov process, the correlation between two samples decreases exponentially as the distance (or lag) time between the samples increases. For a Process showing longer range dependence the correlation decreases more slowly." Thus, if the autocorrelation function decays more quickly than the exponential function then that is some evidence of randomness. If the autocorrelation function is approximately exponential, then that is evidence that the autocorrelation behaviour of the studied parameter can be captured with a Markov model. If the autocorrelation function decays much more slowly than the exponential function then that is evidence of a more complex, long rang dependence.

A valuable tool in the study of autocorrelation is the *autocorrelation plot*, also known as *correlogram*, a graphical representation of statistical correlation. In time series analysis, this plot shows the sample correlation r_h versus the time lag h. If cross-correlation is used, the result is called a cross-correlogram. The correlogram is a commonly used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero.

In the context of the present work, autocorrelations should be near-zero for randomness; if the analyst does not check for randomness, then the validity of many of the statistical conclusions becomes suspect. The correlogram is an excellent way of checking for such randomness. The example in Figure 3.2 shows the autocorrelogram for a series resulting from adding a periodic series with an uniform random sequence. Must be noted that the first vertical line in each autocorrelation plot correspond to $lag = 0$, in other words, the autocorrelation of the time-series with itself, which is always 1.

In the case of a data set that shows strong signs of autocorrelation, it would be advisable to explore a different modelling approach other than the microscale one. Some examples of such analysis based on time-series can be found in [CB07] and [CBA07]. Time-series modelling is out of the scope of the present work and it is not covered by the proposed analysis pipeline. On the other hand, if there is enough evidence about the randomness of the data set it will be possible to move to the next step of the pipeline.

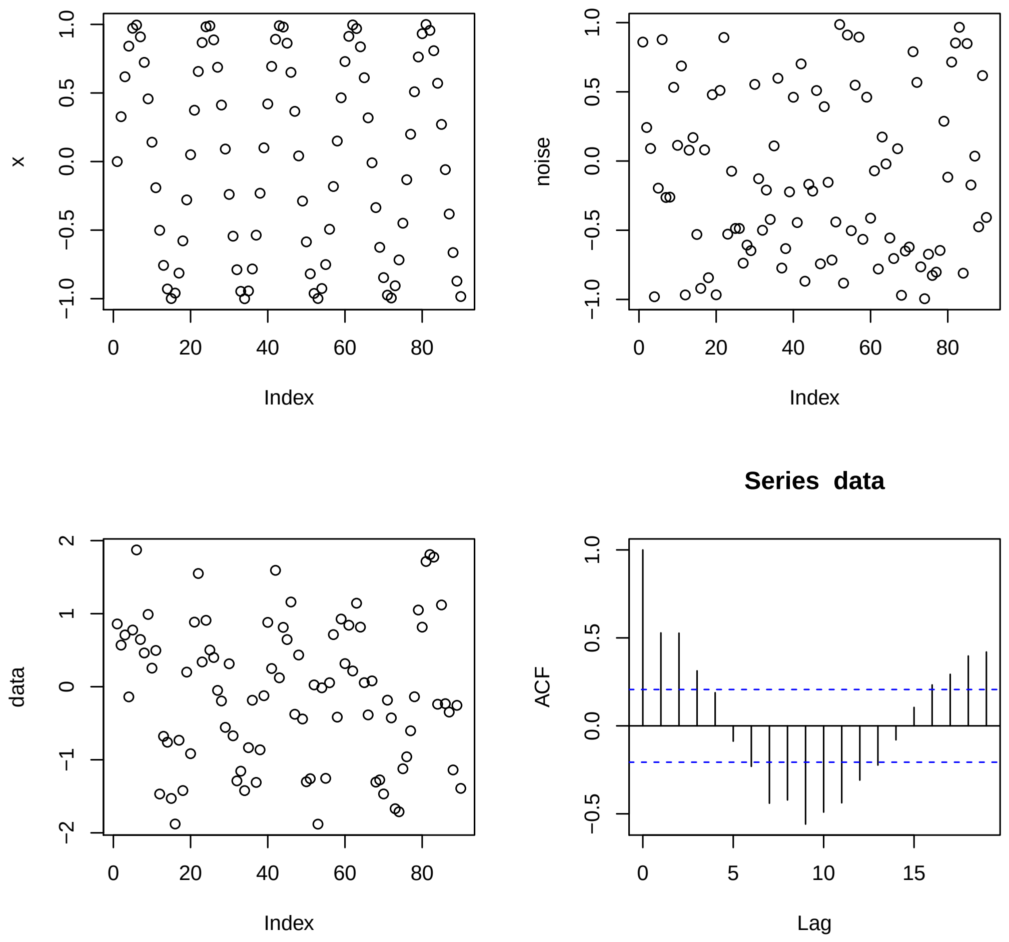


Fig. 3.2.: ACF for uniform random sequence plus periodic series

3.6 Selection of a probability distribution

The *Selection of a probability distribution* step aims to select a probability distribution to model the data set based on the information provided by the preview step and once the autocorrelation study determines it is suitable to further proceed with the pipeline. Choosing a family of theoretical probability distribution will be performed based on the shape displayed by the ECDF associated to the experimental data. This step partly relies on the previous experience of the researcher to identify the similarities between ECDF and theoretical CDF. This makes this step to require human intervention, initially limiting the possibilities to fully automate this part of the pipeline. The first screening in the distribution selection is determining the discrete or continuous nature of the variable associated to the data. There are also cases where a split distribution can be the most adequate choice.

A *discrete probability distribution* is associated to a discrete random variable, a variable that can only assume a finite or countable infinite number of values. The CDF for a discrete probability distribution increases only by jump discontinuities, remaining constant between those jumps. The points where jumps occur are the values which the random variable may take. An ECDF resembling a *stair* is a strong indication that the empirical data may follow a discrete probability distribution. Some well known and studied discrete distributions are:

- Poisson distribution
- Bernoulli distribution
- Binomial distribution
- Geometric distribution
- Negative binomial distribution

A *continuous probability distribution* is a distribution which CDF is continuous. Most often they are generated by having a probability density function (PDF). Some well known continuous probability distributions are:

- Normal distribution
- Exponential distribution
- Gamma distribution
- Weibull distribution
- Uniform distribution
- Chi-square distribution

Due to their extensive use in the modelling literature enumerated in Section 3.1, the following subsections contain a brief description and relevant formulas about the *exponential* and *Weibull* distributions, both continuous probability distributions part of the *exponential family* of distributions.

When evaluating a given ECDF, the researcher may find that it can be divided into different intervals where the shape of the function resembles to different theoretical probability distributions. In this case it is possible to define a *split probability distribution*. This kind of distribution comes handy when a part of the ECDF greatly diverges from the fitting CDF. On the other hand, abusing the splitting and creating a function composed by too many fragments can be a clumsy approach to distribution fitting [Bor00]. Equation (3.5) shows an example of split probability distribution.

$$F(x;\lambda) = \begin{cases} 1 \quad , & x > 0.5 \\ 1-e^{-\lambda x} \quad , & 0.5 \geq x \geq 0 \\ 0 \quad , & x < 0 \end{cases} \tag{3.5}$$

By the end of this step, the researcher will have one or several theoretical probability distributions that will serve as candidates for the next step in the pipeline.

3.6.1 Exponential distribution

The *exponential distribution* is the probability distribution of the time between events in a Poisson point process, i.e. a process in which events occur continuously and independently at a constant average rate. It is a particular case of the gamma distribution. It is the continuous analogue of the geometric distribution, and it has the key property of being memoryless. The PDF of a exponential random variable is described in Equation 3.6, where $\lambda > 0$ is the *rate parameter* of the distribution. The CDF for the exponential distribution is described in Equation 3.7. The mean $E[X]$ and variance $Var[X]$ are described in Equation 3.8.

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{3.6}$$

$$F(x;\lambda) = \begin{cases} 1-e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{3.7}$$

$$\begin{aligned} E[X] &= \frac{1}{\lambda} \\ Var[X] &= \frac{1}{\lambda^2} \end{aligned} \tag{3.8}$$

3.6.2 Weibull distribution

The *Weibull distribution* is a continuous probability distribution named after Swedish mathematician Waloddi Weibull. The PDF of a Weibull random variable is described in Equation 3.9, where $\kappa > 0$ is the *shape parameter* and $\lambda > 0$ is the *scale parameter* of the distribution. The CDF for the Weibull distribution is described in Equation 3.10. The mean $E[X]$ and variance $Var[X]$ are described in Equation 3.11. The Weibull distribution is related to a number of other probability distributions; in particular, it interpolates between the exponential distribution ($\kappa = 1$) and the Rayleigh distribution ($\kappa = 2$ and $\lambda = \sqrt{2\sigma}$).

$$f(x;\kappa;\lambda) \quad = \quad \begin{cases} \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^{\kappa}} & x \quad \geq 0, \\ 0 & x \quad < 0. \end{cases} \tag{3.9}$$

$$F(x;\kappa;\lambda) = 1 - e^{-(x/\lambda)^{\kappa}} \tag{3.10}$$

$$\begin{aligned} E[X] &= \lambda\Gamma\left(1 + \frac{1}{\kappa}\right) \\ Var[X] &= \lambda^2\left[\Gamma\left(1 + \frac{2}{\kappa}\right) - \left(\Gamma\left(1 + \frac{1}{\kappa}\right)\right)^2\right] \end{aligned} \tag{3.11}$$

3.7 Probability distribution fitting

The *Probability distribution fitting* step aims to provide an analytical expression for the model elaborated during the previous steps, as well as determining the values for the model parameters, using as input the theoretical models chosen in the previous step and trying to minimize the statistical discrepancy between the experimental data and the proposed theoretical distributions.

Methods for probability distribution fitting can be classified between *parametric* and *regression* methods. The parametric methods are those where the parameters of the distribution are calculated from the data series. Some examples are the *method of moments* [BS06], *maximum spacing estimation* [Ran84], *method of L-moments* [Hos90] and *maximum likelihood method* [Ald97]. On the other hand there are the regression methods [Oos94], which use a transformation of the cumulative distribution function so that a linear relation is found between the cumulative probability and the values of the data, which may also need to be transformed, depending on the selected probability distribution.

The literature about microscale modelling of network traffic makes an extensive usage of the MLE starting with Bangun et al. [BD00] and Borella [Bor00], and following with the many publications following the guidelines established in these works and enumerated in Section 3.1. Thus, MLE will be the preferred fitting method in the present work.

The MLE [Le 79] [PH94] method allows to estimate the parameters of an statistical model given a data set of observations. The method takes a data sample or population and a statistical model, and calculates the parameter values for this model that maximize the fitting in relation to the sample. The principle of MLE states that the desired probability distribution is the one that makes the observed data *most likely*, which translates into finding the parameter vector that maximizes the likelihood function $L(w|y)$. For example, MLE applied to the exponential distribution aims to determine the value of exponential parameter *rate* (λ), which dictates the behaviour of the distribution, by using as likelihood function $L(w|y)$ the *least squares* function. Thus, λ is calculated by minimising the summation of the square of the distances between the empirical and theoretical values [Myu03].

By the end of this step, the fitting models will count with a well defined analytical expression and a set of values for the corresponding model parameters, ready for being evaluated for later statistical discrepancy.

3.8 Statistical Discrepancy

The *Statistical Discrepancy* step aims to evaluate the GoF of the models generated in the previous step. This can help to determine which is the best fitting model between several candidates. The need to measure statistical discrepancy derives from the fact that a set of observations may be described by various analytical models with different degrees of accuracy, so it is necessary to have an objective criteria to evaluate how close and accurately the data is described by a proposed model. The

GoF of a statistical model gives a measurement of how well a model describes a set of observations by summarising the discrepancy between the observed data and the expected values according to the proposed model.

There are multiple methods aimed to determine if a given distribution properly describes the distribution of a data set. These methods have different characteristics which made them more suited to certain cases than others: some of them are aimed to determine the normality of a data set, others are more sensible to extreme values or heavy tailed distributions, etc.

There are several methods to measure statistical discrepancy and GoF, some of them just provide a visual representation that can be intuitively interpreted, others provide numerical metrics that allow direct comparison of GoF between models for a given data set. The methods used to measure GoF in the present work have been chosen from the literature described in Section 3.1. The following subsections contain an overview of each of them.

3.8.1 Q-Q plots

A Q-Q plot, where Q stands for *quantile*, is a probability plot which provides a graphical method for comparing two probability distributions by plotting their quantiles against each other. A Q-Q plot allows to compare the shapes of distributions by providing a graphical view of the similarity of their properties. Thus, it is a tool that can be used to compare collections of data or theoretical distributions [Cha+83], providing a more complete tool for CDF comparison, allowing to determine other differences like shape, skew, weight of the tails and so on. Q-Q plots are used in the present study to compare data sets of experimental observations to the proposed theoretical models. [Bor00] is one of the early examples of the video game network modelling using Q-Q to assets statistical discrepancy, others such as [PKK05], [DPV05] or [CB07] also followed this approach.

The first step to create a Q-Q plot is defining the number of intervals (quantiles) in which the distributions will divided into. Every point (x, y) on the plot is composed by one of the quantiles of the first distribution (x-coordinate) and the equivalent quantile of the second distribution (y-coordinate). The result is a parametric curve, being the parameter the number of intervals for the quantile. The points plotted in a Q–Q plot are always non-decreasing when viewed from left to right. If the two distributions being compared are identical, the Q–Q plot follows the 45º line $y = x$. If the two distributions agree after linearly transforming the values in one of the distributions, then the Q–Q plot follows some line, but not necessarily the line $y = x$. If the general trend of the Q–Q plot is flatter than the line $y = x$, the distribution plotted on the horizontal axis is more dispersed than the distribution plotted on the vertical axis. Conversely, if the general trend of the Q–Q plot is steeper than the line $y = x$, the distribution plotted on the vertical axis is more dispersed than the distribution plotted on the horizontal axis. Q–Q plots are often *S* shaped which indicates that one of the distributions is more skewed than the other, or that one of the distributions has heavier tails than the other.

Table 3.1 shows the quantiles (10 intervals) for two data sets. The empirical quantiles correspond to a data set of 100 observations generated by a random generation function for the normal distribution $\mathcal{N}(\mu = 0, \sigma = 1)$. The theoretical quantiles were calculated using the quantile function for $\mathcal{N}(\mu = 0, \sigma = 1)$. The Figure 3.3 shows the corresponding Q-Q plot. The random numbers were generated using a generator for normal distribution, so the resulting points on the plot are close to the $y = x$ line.

Tab. 3.1.: Quantiles for Q-Q example

Quantile	Empirical	Theoretical
10%	-1.4219	-1.2816
20%	-0.9321	-0.8416
30%	-0.5548	-0.5244
40%	-0.2060	-0.2533
50%	-0.0973	0.0000
60%	0.1049	0.2533
70%	0.4110	0.5244
80%	0.6374	0.8416
90%	1.0499	1.2816

3.8.2 λ^2 and $\hat{\lambda}^2$ for discrepancy measurement

The λ^2 Discrepancy Measure quantifies the GoF between a data set and a given analytical model. This metric was introduced in [PJ90] and it has been widely used in multiple works dealing with traffic network analysis, such as [Bor00], [Pax93], [LHR02], [DPV05] or [BA06].

The λ^2 distribution is based on the chi-square (χ^2) goodness of fit test in an attempt to normalise its results and obtain a quantitative measurement of discrepancy. One of its main advantages over χ^2 is the possibility to compare two λ^2 values from different pairs of data sets and theoretical distributions and determine which pair show the best GoF. χ^2 formula is shown Equation (3.12), where N is the number of bins in which the empirical observations has been divided, E_i is the number of

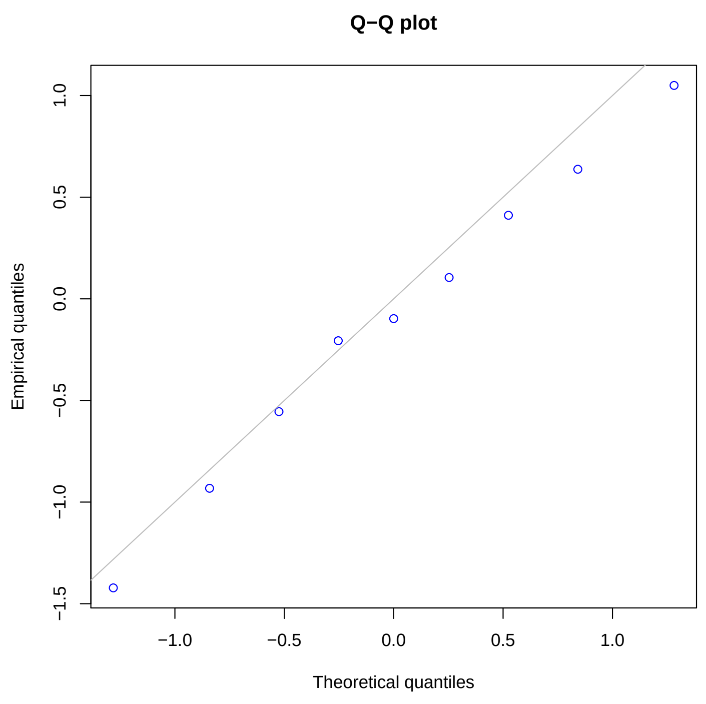


Fig. 3.3.: Q-Q plot, normal sample vs. normal quantiles

observations in the $i-th$ bin and T_i the number of observations expected for that bin according to the given theoretical distribution.

$$\chi^2 = \sum_{i}^{N} \frac{(E_i - T_i)^2}{T_i} \tag{3.12}$$

The original formula for λ^2 proposed by Pederson and Johnson in [PJ90] is shown in (3.13), where n is the number of observations in the data set and df is the number of degrees of freedom of the test. The formula for parameter K is shown in Equation (3.14), where N is the number of bins, E_i is the number of observations in the $i-th$ bin and T_i the expected number of observations for that same bin.

$$\lambda^2 = \frac{\chi^2 - K - df}{n-1} \tag{3.13}$$

$$K = \sum_{i}^{N} \frac{E_i - T_i}{T_i} \tag{3.14}$$

Previous works addressing λ^2 state that the measurement is not reliable when the number of bins is too big. Trying to fit very heavy-tailed distributions generate very large λ^2 values. In this case there are two possible approaches: [Pax93] proposes to log-transform the data to reduce the tail and number of bins. On the other hand, [Bor00] opts for truncating the upper tail when it prevents a good fit.

Negative or close to 0 λ^2 values indicate a high degree of fitting between the data sample and the fitting distribution. On the other hand, λ^2 values of several orders of magnitude are a sign of lack of fitting. Even so, λ^2 cannot be considered an absolute fitting metric, but a tool to compare the fitting of samples from a common population. Thus, a λ^2 value in the order of thousands indicates a worse fitting that a value close to 0, but there is not necessarily a direct relationship between the absolute value of λ^2 and the goodness of the fit.

The λ^2 calculation process may encounter divide-by-zero situations when the expected number of observations T_i is 0 for the given theoretical distribution. To avoid such a situation, Borella [Bor00] proposed an alternative definition, $\hat{\lambda}^2$ which inherits the advantages of λ^2 while avoiding divide-by-zero situations. To solve the opposite situation where the empirical observations E_i are zero for a given bin, this bin is just ignored and no computed.

The formula for $\hat{\lambda}^2$ metric in Equation (3.15) relies on $\hat{\chi}^2$ (Equation (3.16)) and $\hat{K}$ (Equation (3.17)), reformulations of χ^2 and K respectively.

$$\hat{\lambda}^2 = \frac{\hat{\chi}^2 - \hat{K} - df}{n-1} \tag{3.15}$$

$$\hat{\chi}^2 = \sum_{i=1}^{N} \frac{(T_i - E_i)^2}{E_i} \tag{3.16}$$

$$\hat{K} = \sum_{i=1}^{N} \frac{T_i - E_i}{E_i} \tag{3.17}$$

Both χ^2 and λ^2 as well as their variants $\hat{\chi}^2$ and $\hat{\lambda}^2$ require a binning method for continuous distributions. [PJ90] includes a discussion about some of the available methods. This work uses the binning technique introduced in [Sco79] and also employed [Pax93] and [Bor00], based on choosing an optimal number of fixed-size bins of width w given by Equation (3.18) where σ is the empirical standard deviation and n is the number of observations in the data set.

$$w = 3.49\sigma n^{-1/3} \tag{3.18}$$

Fixed-width bins have been chosen due to the latter case, a long-tailed distribution will often have its entire tail contained into a single bin, which would provide a too optimistic estimation about the GoF between the two compared distributions. It is important to understand the impact of bin size on the GoF discrepancy metrics. χ^2 is very sensitive to bin choice while λ^2 is relatively insensitive, only fluctuation in a factor of 2 or 3 if the bin size is close to the optimal value w.

$\hat{\lambda}^2$ has been chosen as one of the quantitative GoF metrics in the present study regarding its capacity to quantitatively compare the GoF of two different models for a given data sample or population. Appendix Section C includes the implementation for λ^2 and $\hat{\lambda}^2$ in R language used in the data analysis performed for the present study.

3.9 Summary

The present chapter includes an overview of statistical methods used in the literature for network traffic modelling of video games, choosing those compatibles with the study of NVEs. The chapter deals with:

- Overview of the literature focused on modelling of network traffic for video games that share characteristics with the definition of NVE, those sharing a simulation over a network environment such as FPS, MMORPG or certain RTS titles. The approach chosen for the modelling of NVE network traffic has been the microscale one, popular in the literature dealing with the previous genres, which is based on modelling the IAT and PS of the traffic generated by the video game / NVE.
- Definition of the basic workflow for an analysis pipeline for statistical analysis following the microscale modelling approach. Such a pipeline can be automate up to certain degree, but some of the steps require the evaluation of the researcher performing it, i.e. choosing the best candidate models to proceed with the model fitting process.
- Guidelines for the initial data filtering plus considerations related to the management of outliers and distribution tail. The filtering will be performed depending of the type of traffic to be studied.

- Data preview to provide information to the researcher applying the pipeline to make technical decisions such as the nature and type of models to evaluate to generate the final one. Data preview heavily relies on the graphical representation of the ECDF for the input data.
- Study of autocorrelation as a necessary test to determine the randomness of the data input and the suitability of using probabilistic models to describe it. A strongly autocorrelated data traffic may require using different models such as time-series or Markov ones, out of the scope of the proposed pipeline.
- Selection of the probabilistic models that will be shaped and optimised into the final ones. The data previews generated during early steps will help to determine the nature of the data model (discrete vs. continuous) or the necessity of using a split model. The candidate models will get values for their parameters using the MLE method. In both the literature and the present study the exponential and Weibull model has been widely applied, thus including an overview of their main characteristics.
- Study of statistical discrepancy to determine the degree of fitness or GoF of the generated models. The two chosen methods are Q-Q plots, to provide a visual and intuitive measurement of the GoF and the family of λ^2, a refinement of χ^2 with desirables properties for data collections with similar probabilistic nature than NVE network traffic and similar video games.

4 Analysis and modelling of NVE network traffic: Open Wonderland

In other words, the given time of arrival is the one moment of time at which it is impossible that any member of the party will arrive.

Douglas Adams, *Life, the Universe and Everything*

This chapter aims to apply the pipeline described in Chapter 3 to the network traffic generated by Open Wonderland (OWL), the networked virtual environment (NVE) system chosen as case study as a representative example of NVE. This pipeline has been designed based on previous works on video game network traffic modelling, and taking into consideration the special characteristics of NVEs to determine the most appropriate approach to the NVE network traffic modelling. The goal of applying the analysis pipeline to the OWL network traffic is to prove its suitability to generate accurate models that can be used in later activities such as simulation, network traffic extrapolation or sizing network infrastructures for NVE systems.

This chapter relies on the analysis pipeline described in chapter 3 to analyse the empirical network traffic generated by the NVE subject of study in the present work, OWL. This traffic has been generated in the testbeds Open Wonderland TestBed 1 (OWTB1) and 2 Open Wonderland TestBed 2 (OWTB2), both described in chapter 2. The guidelines used to perform the OWL testing gaming sessions and capture the empirical network traffic data have been described in 2.3.3. Each section details the results obtained for OWTB1 and OWTB2 described in 2.3.1 and 2.3.2 respectively.

The chapter narrows the scope to traffic associated to object synchronisation within OWL sessions. Audio traffic has been analysed and modelled in previous papers derived from the present work, showing a highly periodic packet rate and almost constant bandwidth. Due this simplicity and the space constraints, it has been left out of the scope of the present document. Details about the audio traffic study can be found in the article [Fon+12b].

The chapter also provides a description of the variables that define it at microscale model level, associated to inter-arrival time (IAT) and packet size (PS) respectively. The modelling of each variable is performed by applying the analysis pipeline previously defined, and the details about the modelling process are included. After applying the pipeline for each of the microscale variables, the resulting statistical models are presented, including a range of valid values of their parameters.

4.1 Object synchronisation in Open Wonderland

Objects within the OWL virtual world have a set of attributes that can have different values: for instance, coordinates in the virtual world, velocity (for moving objects), etc. The state of each object (or *cell*) is defined as the values of these attributes at a given time. In Wonderland, the server keeps a copy of all the world data, with data about the cells stored in XML files. When a client connects to Wonderland, it obtains from the server the information about the visible objects (cells). These objects can be classified as static, with fixed attributes that do not vary in time, as it is the case of a mountain or a building; or dynamic, with attributes that can vary in time, like an avatar moving from one region to another [Lui01].

Within the shared simulation provided by OWL, every client should have a consistent view of the virtual world, and therefore a mechanism must exist to ensure synchronisation between clients. When an object moves in this virtual world all clients that want to view the dynamic object must provide the same sequence of state changes. When something changes within OWL, such as the position of an object, its associated data are updated on the server and then forwarded to all the clients, which then update their own local views of the world. For instance, if a client moves their avatar, the client notifies the server and sends the new state of the cell to the server. Then, the server sends the new state of this object to all other clients, which then update their copies of the cell [Fouf].

The object synchronisation within OWL is performed over TCP protocol. While other game genres, such as first-person shooter (FPS), sacrifice the reliability of TCP over the speed of UDP, OWL has chosen to use TCP, which ensures the object updates are properly propagated and confirmed over the networked simulation. This TCP traffic is exchanged by default through port 1139 [Com].

The inspection of OWL network traffic at packet level using Wireshark [Tea19b] reveals that the typical OWL network packet exchange associated to Object Synchronisation happens as follows:

1. The OWL client sends a *MOVE_REQ* message to the server when the position of an object changes.
2. The server propagates the update above by sending a *MOVED* message to every other client, one per client.
3. Every client responds to the update with an ACK message. The server does not send any other MOVED message to a client until this responds with an ACK to the previous update.

Although these *MOVE_REQ* messages are sent every time the object moves, in principle they are limited by the *MoveableComponent*. According to the Open Wonderland Project Forum [Foue], this limit is 5 updates per second, although this statement could not be confirmed in our experiments as we will discuss later on.

4.2 Microscale modelling and analysis pipeline

The analysis pipeline for NVE network traffic modelling defined in Chapter 3 follows a microscale approach that aims to fully characterise the intra-session dynamics of the NVE network traffic by means of two variables, the IAT (or inter-departure time (IDT), depending on the observation point of view) and the PS. The empirical network traffic for modelling has been captured on the server side, as described in section 2.3.4. Moreover, this study focuses on the *outgoing client traffic / outgoing server traffic*. Thus, the text addresses the time between consecutive incoming packets that arrive to the OWL server from the different clients as *Inter-Arrival Time (IAT)*, in the same way, *Packet Size* (PS) refers to the packet size of the incoming network traffic relative to the OWL server.

4.3 Data filtering criteria

The object synchronisation traffic generated by OWL is composed of TCP packets that can be identified by certain tokens contained in their payload. Specifically, the update request contains the plain text *MOVE_REQUEST* token, while their confirmation packets contain the text token *MOVED*.

Network traffic captured during the first and last minute of each session for both testbeds was discarded to avoid patterns generated by the login and logout process. All the traffic was captured using *Wireshark*[Tea19b], an open-source packet analyzer, as well as *tshark*[Tea19a], its CLI counterpart and better suited for scripting and automation of the packet capture. Further details about the packet capture process and parametrization can be found in Appendix B, Section B.1.

Once the OWL object synchronisation packets have been isolated from the raw network capture, the data set is ready for calculating the IAT for consecutive packets of the same type. This data set will be used by the analysis pipeline to provide a model that describe its behaviour.

4.4 Object synchronisation inter-arrival time

The term object synchronisation inter-arrival time (OSIAT) defined in the present work refers to the time between consecutive object synchronization packets arriving to a given host, in this case, the server. The OSIAT is one of the two variables used in the microscale modelling approach for modelling NVE network traffic. The following sections describe how the OSIAT has been measured and analysed to produce statistical models.

4.4.1 Data preview

The preliminary study of OSIAT values has consisted in a preview of the empirical cumulative distribution function (ECDF) for the object synchronization traffic received from each client within the session. This preview shows all the incoming client traffic OSIAT follows a similar pattern. Figure 4.1 shows the ECDF plots belonging to *Client 2* from the 2-client session and *Client 3* from the 5-client session (see Table 2.1 and Section 2.3.1 for further details on the client's hardware and software).

Further ECDF plots for OSIAT traffic can be found in Appendix D, specifically the Figures D.1, D.2 and D.3 which show the ECDF plots for OSIAT values for clients engaged in the 2, 3 and 5-player sessions respectively.

ECDF plots for OWTB1 clients, as displayed in Figure 4.1, show a continuous nature in the values for IAT, as well as two differentiated sections divided at $0.5\,\mathrm{s}$. Values below $0.5\,\mathrm{s}$ seem to follow an exponential or Weibull distribution. This exponential nature is plausible if we assume the fact that outgoing client traffic can be considered as a Poisson Process where the waiting time between consecutive packets follows an exponential probability distribution [Ros95].

OSIAT values greater than $0.5\,\mathrm{s}$ represent a small fraction of the total and varies between clients and sessions. Although the ECDF curves have very similar shapes, they are not completely the same. Further tests within the testbeds showed that IAT distribution depends on session parameters such as user activity, number of players or client resources. Section 4.4.3 discusses the nature of these values and their relationship with user activity during the gaming session.

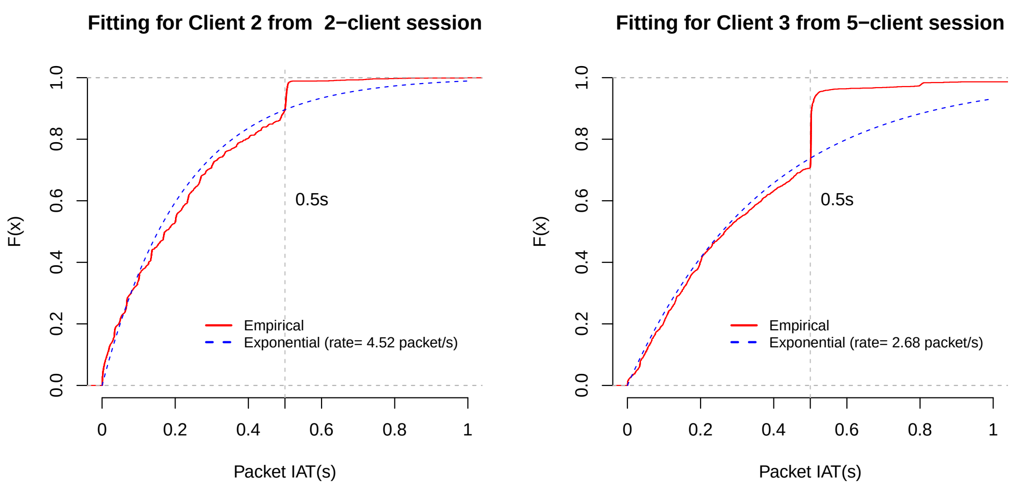


Fig. 4.1.: ECDF for OSIAT and CDF for exponential distribution, OWTB1

There is a probability saturation around $0.5\,\mathrm{s}$ with an important percentage of OSIAT values concentrated around a narrow range centered on this value. This percentage varies between clients; in Figure D.2 the ECDF for OSIAT values generated by *Client 1* has a lower step around $0.5\,\mathrm{s}$ than that of *Client 3*. In both cases, values greater than $0.5\,\mathrm{s}$ are relatively unlike and their frequency also varies between clients.

According to the information published in the OWL documentation [Proc], clients limit the synchronisation traffic to avoid rates greater than 5 packets per second. This would imply a minimum OSIAT value of $0.2\,\mathrm{s}$. However, it was found a big incidence of values around $0.5\,\mathrm{s}$. A possible explanation for this concentration of OSIAT values around $0.5\,\mathrm{s}$ is object synchronisation packets follow an exponential-like distribution in optimal conditions, but when there is a saturation of synchronisation packets, the OSIAT values fall into the worst-case scenario for the system, this is $0.5\,\mathrm{s}$.

Each ECDF curve in Figure 4.1 is represented by a solid line, accompanied by a dashed one corresponding to the cumulative distribution function (CDF) $F(x)$ for an exponential distribution calculated by maximum likelihood estimation (MLE). The visual inspection of the fitting suggests an exponential nature for the OSIAT values below $0.5\,\mathrm{s}$. The rate parameter λ determines the exponential behaviour and can be interpreted in the IAT context as the number of packets per second generated by an OWL client. Although this exponential functions are a rudimentary approximation, they support the hypothesis of an exponential distribution of part of the OSIAT values.

The OSIAT values for OWTB2 clients display similar ECDF plots to those from OWTB1. Figures D.4, D.5, D.6 and D.7 show the ECDF plots for OSIAT from 2, 3, 5 and 8-players sessions performed on OWTB2. OSIAT distribution is consistent over the two testbeds.

Figure 4.1 suggests that pure exponential models can provide some degree of fitting for the OSIAT empirical data but the ECDF tails and region around $0.5\,\mathrm{s}$ clearly diverges from the exponential CDFs calculated using MLE. This behaviour can be modelled using a *split model* [Bor00] or *split distribution*, an analytical expressions that divides the data range in several regions, providing a different model to describe each one. While assembling several models into one single split distribution may produce a clumsy models, the OSIAT values only show three well defined regions, which allows the definition of a relatively simple split distribution. Thus, an analytical expression for OSIAT would be initially defined by the following parameters:

- a threshold t_{thres} value delimiting the two areas following different probability distributions.
- an exponential or Weibull distribution modelling the probability for OSIAT values between 0 and t_{thres}.
- a probability saturation located in the t_{thres} value.

Thus, the data preview for OSIAT reveals a continuous variable with exponential-like behaviour in the interval between 0 and $0.5\,\mathrm{s}$, followed by a saturation of probability around $0.5\,\mathrm{s}$ and a small distribution tail. The modelling strategy that will be further developed in later sections will take into account the dual nature of the OSIAT values, first modelling the exponential-like behaviour of values less than $0.5\,\mathrm{s}$ and then the probability pulse around $0.5\,\mathrm{s}$ itself. A split model will be suitable to capture this dual behaviour.

4.4.2 Autocorrelation

The study of autocorrelation in a data set may help to discover inherent dependencies and determine the modelling strategy. In the case of OSIAT, Figure 4.2 shows the autocorrelation function (ACF) for the 2-client gaming session from OWTB1. Further ACF figures are included in Appendix D.

Attending to these ACF plots, OWTB1 OSIAT values show a very low degree of autocorrelation. The 2-client session ACFs in Figure D.9 do not show significant autocorrelation, while 3-client sessions in Figure D.10, *Client 2* shows slightly higher values than the rest of the clients. This divergence is probably due to a combination of factors such as hardware, operating system configuration or specific software versions running on the client. ACFs for 5-client session in Figure D.11 show similar negligible autocorrelation for OSIAT values, however some clients show a slightly higher values for specific lag values, but they cannot be considered statistically significant, being all of them lower than 0.2.

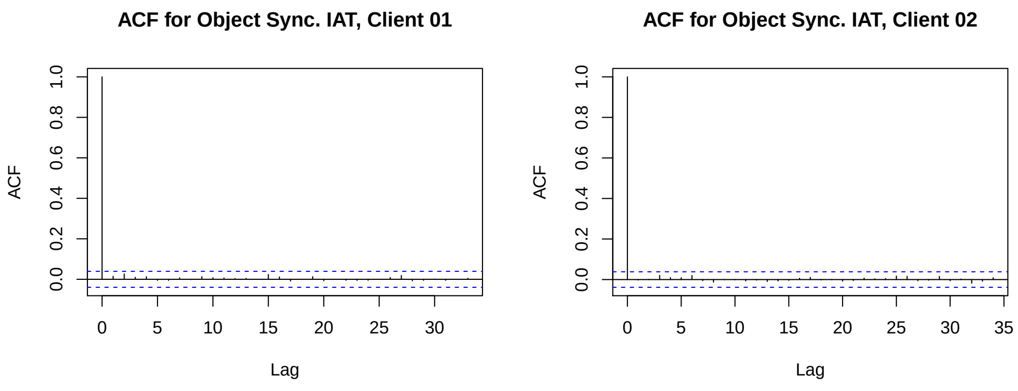


Fig. 4.2.: ACF for OSIAT values, 2-player session, OWTB1

Regarding autocorrelation in the OWTB2 sessions, the Figure 4.3 shows the ACF plots for the 2-client sessions on this testbed. Further ACF plots for 2, 3, 5, 8 and 10-client sessions performed on OWTB2 are included in Appendix D.

While all these figures do not provide strong evidences of autocorrelation, there is some degree of inverse autocorrelation (values between -2 and -3) for $lag = 5$ in all the sessions, with the exception of the single user one. There is also a small peak of direct autocorrelation (values lower than 2) for $lag = 10$. These peaks of inverse and direct autocorrelation repeat themselves in steps of 5 lag units until becoming negligible. This is likely a side-effect of using scripts to automate and maximize avatar activity in OWTB2 as described in Section 2.3.2, which has pushed the rate of events to its limit, creating some ripples of autocorrelation and inverse autocorrelation, but not strong enough to classify the traffic as strongly autocorrelated.

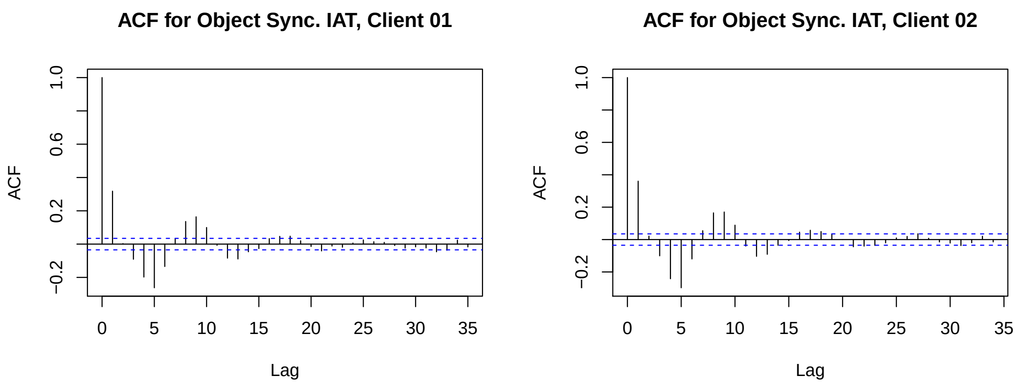


Fig. 4.3.: ACF for OSIAT values, 2-player session, OWTB2

For both testbeds, the hints of autocorrelation for lag values between 1 and 5 suggest small burst of object synchronisation packets or the fragmentation of bit update packets for later transmission on the communication channel. This lack of autocorrelation is a good indicator of randomness and the suitability of the proposed analytical pipeline to model the OSIAT values as a random variable.

4.4.3 User activity

The ECDF plots for OSIAT values in Section 4.4.1 showed rather lightweight distribution tails, with a low percentage of values above $0.5\,\mathrm{s}$ and also how these percentages varied from one to another client within the same session. From the OWL documentation it can be inferred that there is a relationship between the user activity and the percentage of high OSIAT values [Fouf]. Thus, inactivity periods for the avatar would imply the absence of object synchronisation traffic and therefore greater time lapses between consecutive synchronisation packets, which translates into greater IAT values.

Determining more accurately the impact of the client activity on the distribution of OSIAT values requires a controlled gaming session. For this reason two extra single-player sessions were performed in OWTB1 with the goal of isolating the effect of avatar activity rate on the object synchronisation traffic and its associated IAT values. In one of the sessions the avatar performed a high rate of interaction within the gaming session (constant movement and audio transmission), while a second gaming session was performed by an avatar staying inactive for long periods of time.

Figure 4.4 shows the respective ECDF plots for the active and inactive clients. The plots suggest that in presence of a hight rates of user activity and associated traffic, the object synchronisation packets are delivered with a maximum IAT of $0.5\,\mathrm{s}$. In this scenario, the few tail values greater than $0.5\,\mathrm{s}$ are more likely due to the saturation of the clients or server rather than the lack of synchronisation packets. On the other hand, during the periods of player inactivity OSIAT values increased, likely due to the lack of updates involving greater times between the synchronisation packets.

Table 4.1 gives a quantitative insight for the distribution tails in Figure 4.4 by providing from 89^{th} to 99^{th} percentiles for Object Synchronisation traffic from the active/inactive sessions. According to these figures, the *inactive player* has a noticeable percentage of OSIAT values equal or greater than $0.5\,\mathrm{s}$ ($11\,\%$), while the active player has fewer values greater than $0.5\,\mathrm{s}$, less than $1\,\%$.

Table 4.2 contains a similar summary about OSIAT tail values for the OWTB1 gaming sessions (2,3 and 5-player sessions respectively). Figures in column $Q_{0.96}$ show that $96\,\%$ of all the OSIAT values were below $0.56\,\mathrm{s}$, while $98\,\%$ were smaller than $1\,\mathrm{s}$.

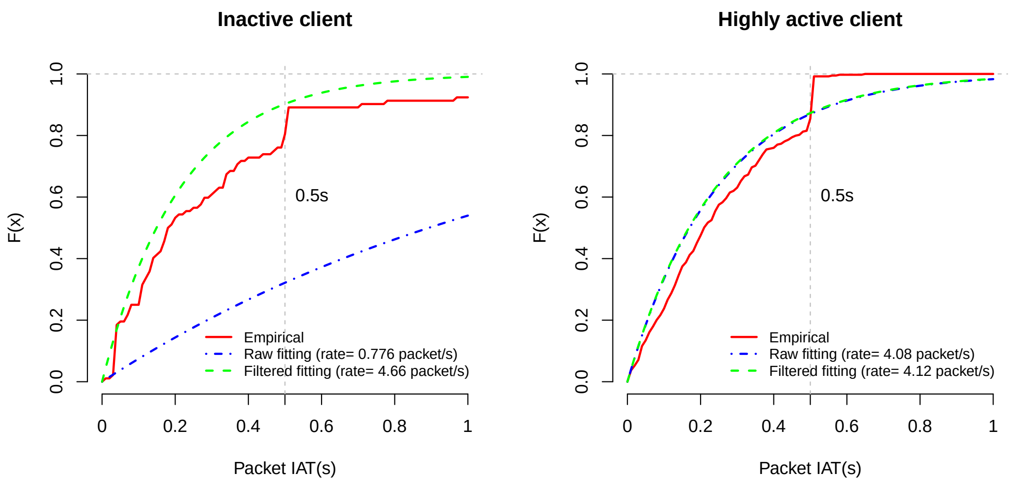


Fig. 4.4.: ECDF and fitting CDF for user activity study sessions, OWTB1

Tab. 4.1.: Quantiles for OSIAT from *active/inactive* testing sessions

Quantile	Inactive client (s)	Active client (s)
$Q_{0.89}$	0.5010	0.5003
$Q_{0.90}$	0.6840	0.5003
$Q_{0.91}$	0.7609	0.5004
$Q_{0.92}$	0.9104	0.5005
$Q_{0.93}$	0.9913	0.5006
$Q_{0.94}$	1.1978	0.5007
$Q_{0.95}$	2.2352	0.5008
$Q_{0.96}$	5.7679	0.5009
$Q_{0.97}$	11.4161	0.5010
$Q_{0.98}$	17.8715	0.5011
$Q_{0.99}$	32.0450	0.5016

Although the distribution of OSIAT values varies between clients, the percentage of *tail values*, those greater than $0.5\,\mathrm{s}$, are consistent with the high user activity with which the players were performing during the OWTB1 sessions.

Tab. 4.2.: Quantiles for OSIAT, OWTB1 sessions

Session	Client	$Q_{0.95}$	$Q_{0.96}$	$Q_{0.97}$	$Q_{0.98}$	$Q_{0.99}$
2-client	01	0.5030	0.5040	0.5080	0.5141	0.6545
	02	0.5038	0.5048	0.5061	0.5081	0.6208
3-client	01	0.5058	0.5072	0.5100	0.5224	0.7631
	02	0.5046	0.5056	0.5073	0.5120	0.6931
	03	0.5028	0.5176	0.6980	0.8027	0.8398
5-client	01	0.5068	0.5098	0.5127	0.6419	0.7600
	02	0.5049	0.5064	0.5082	0.5119	0.6682
	03	0.5211	0.5510	0.7350	0.8051	1.6982
	04	0.5117	0.5139	0.5177	0.6530	0.7644
	05	0.5014	0.5030	0.6203	0.7957	0.8039

Table D.1 in Appendix D contains percentiles from 95^{th} to 99^{th} for OSIAT values for each client and session from OWTB2. Figures show all these OSIAT distributions are even more lightweight tailed than those from OWTB1. This can be explained by replacing human players with scripts controlling the avatars during the experimental gaming sessions. The scripts were programmed to perform at maximum activity ratio, minimising the tail of OSIAT values.

OSIAT tail values in Tables 4.1, 4.2 and D.1 reveal a relationship between the rate of activity displayed by the avatar during the session and the weight of the distribution tail for OSIAT values and the higher the activity rate, the lighter the distribution tail. This relationship will help to make decisions concerning tail and extreme OSIAT values during the modelling step, i.e. minimising the tail would be equivalent to model a user engaged in high rate of activity within the game, where the maximum values for OSIAT are not greater than $0.5\,\mathrm{s}$ than the high rate of synchronisation packets.

A closer look to the user activity in OWL reveals an inverse relationship between the level of user activity, in the sense of translation and interaction within the gaming session, and the size of the distribution tail for OSIAT values. Based on this relationship, it is possible to propose not to model this distribution tail for the sake of simplicity while still obtaining a model that is representative of a gaming case in which the player performs a degree rate of translation and interaction.

4.4.4 Data modelling

In Section 4.4.1 the preview plots pointed to *split distributions* as a viable model approach for OSIAT. Such a split distribution-based model would have a first interval of exponentially distributed for values between 0 and $\approx 0.5\,\mathrm{s}$, a probability saturation around $0.5\,\mathrm{s}$ that generates the step visible in the ECDFs, and finally a tail that comprises a low percentage of OSIAT values, inversely proportional to the activity rate of the user during the gaming session (Section 4.4.4).

The split distribution-based mode for OSIAT values will be defined by the following parameters:

- Parameters that define the exponential-like behaviour of values between 0 and $\approx 0.5\,\mathrm{s}$. They will depend on the theoretical distribution chosen to model this part.
- Threshold, t_{thres}, time value that divides exponential-like region from the probability saturation.
- Weight of the values in the probability saturation region in relation to the whole population, C_{acp}.

OSIAT tail values are left outside the scope of the model for the sake of simplicity and avoiding to produce a too complex split distribution, based on the experimental evidence described in Section 4.4.3 where gaming cases with high rate of activity translate in a negligible tail for OSIAT values. Moreover, the model is aimed to evaluate and size network requirements, and in this context tail values do not play an important role.

Threshold parameter

The threshold parameter t_{thres} parameter defines the boundary between the two sections of the split distribution proposed in Section 4.4.1 for OSIAT values. The ECDF plots in Figures D.1, D.2, D.3, D.4, D.5, D.6, D.7 and D.8 all show a probability saturation around $0.5\,\mathrm{s}$ in the form of a sharp step in the ECDF.

Figure 4.5 shows a detail of the probability density function (PDF) for all the OSIAT values obtained in OWTB1. The plot focuses on a narrow range around $0.5\,\mathrm{s}$. Figure 4.6 shows the same range for the OSIAT values from OWTB2. In this case the values are concentrated in an even narrower interval of x. Both figures highlight the *median*, $\widetilde{x}_{OWTB1} = 0.5010s$ and $\widetilde{x}_{OWTB2} = 0.5011s$ respectively. The median has been chosen as representative statistical parameter of the interval due to its lower sensitivity to extreme values.

Taking into account that the OSIAT values from OWTB1 and OWTB2 are just a representation of the all possible values that can be observed in the OWL traffic, together with the errors inherent to the traffic capture and measurement process, $t_{thres} = 0.5s$ will be used in the present work to determine the boundary between the two parts the OSIAT distribution is divided. Using further decimals could just fall into model overfitting for the given experimental data while not providing any tangible benefit.

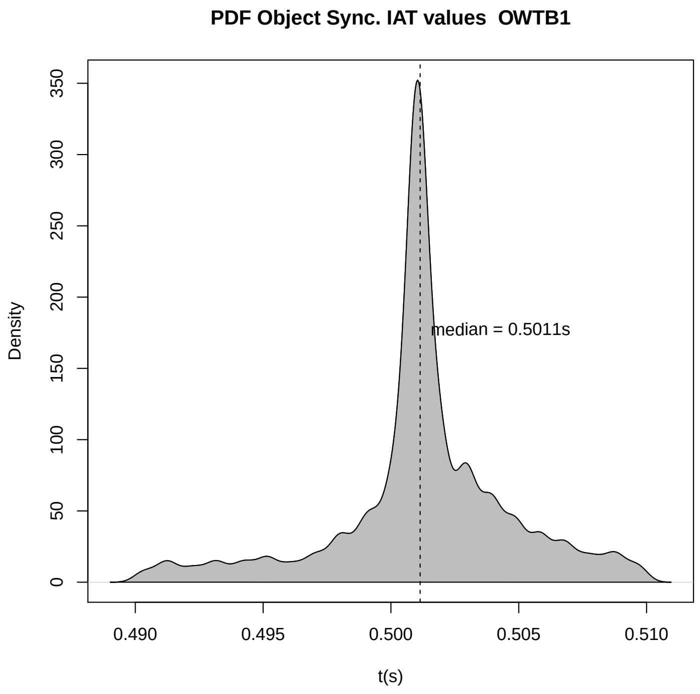


Fig. 4.5.: PDF for the aggregation of OSIAT values from OWTB1

Activity Correction Parameter

The *Activity Correction Parameter*, C_{acp}, aims to weight the exponentially distributed OSIAT values regarding the overall OSIAT distribution. Section 4.4.3 discussed the relationship between the rate of user activity within the gaming session (movement and audio transmission) and the percentage of OSIAT values equal or greater than $0.5\,\mathrm{s}$. The ECDF plots for the user activity experiment showed that high inactivity periods imply heavy tailed OSIAT distributions. Thus, depending on the activity rate displayed during the gaming session and the associated distribution tail, the fitting curve obtained using MLE to model values in interval $[0, t_{thres}]$ will be required to be adjusted.

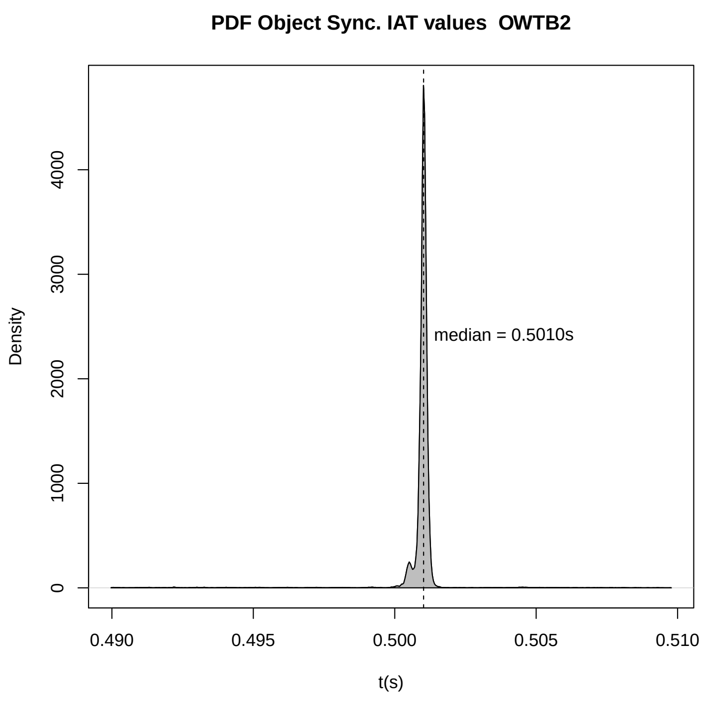


Fig. 4.6.: PDF for the aggregation of OSIAT values from OWTB2

The *client 1* from the 5-client session performed on OWTB1 is used as reference to define the formula for C_{acp}. Figure 4.7 shows the ECDF for the OSIAT values for this client (solid black line). The red dashed line corresponds to the exponential distribution calculated using MLE. It can be appreciated that probability saturation around $0.5\,\mathrm{s}$ diverts the fitting curve from the empirical results. The goal of the C_{acp} is to correct this deviation by escalating the fitting curve.

The scaling factor C_{acp} is calculated by measuring the difference between the ECDF and the fitting curve at $P_U = 0.49s$. The choice of this value correspond to the initial of the interval where the vast majority of OSIAT values around $0.5\,\mathrm{s}$ are concentrated (Section 4.4.4), being a convenient point of union between the two parts of the OSIAT model Figure 4.8 zooms over the area where the studied ECDF and fitting curve meet, highlighting the curve points used to calculate the value of δ. The definition of C_{acp} implies that its value for a given model depends on the fitting curve for interval $[0, t_{thres}]$, at point $P_U = 0.49s$.

The difference between these curves at $P_U = 0.49$ is expressed in Equation (4.1) by δ as the difference between the ECDF value for $0.49\,\mathrm{s}$ and the value $P_{FIT}(P_U, \{P_{rate\_mle}\})$, the cumulative probability of the fitting curve P_{FIT} with parameters $\{P_{rate\_mle}\}$ at point $P_U = 0.49s$. Thus, the C_{acp} is a scale factor described in Equation (4.2). The Figure 4.8 highlights the point $pexp(0.49s, rate = 3.74)$, the value given by the fitting exponential CDF with rate parameter $\lambda = 3.74$ (the $P_{FIT}(X, rate_mle)$ function

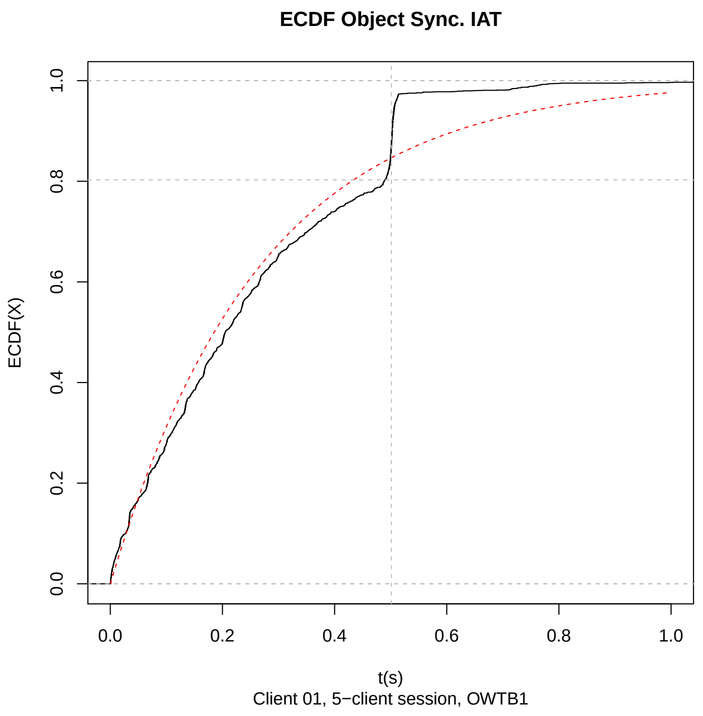


Fig. 4.7.: ECDF and fitting CDF, *Client 1*, *5-client session*, OWTB1

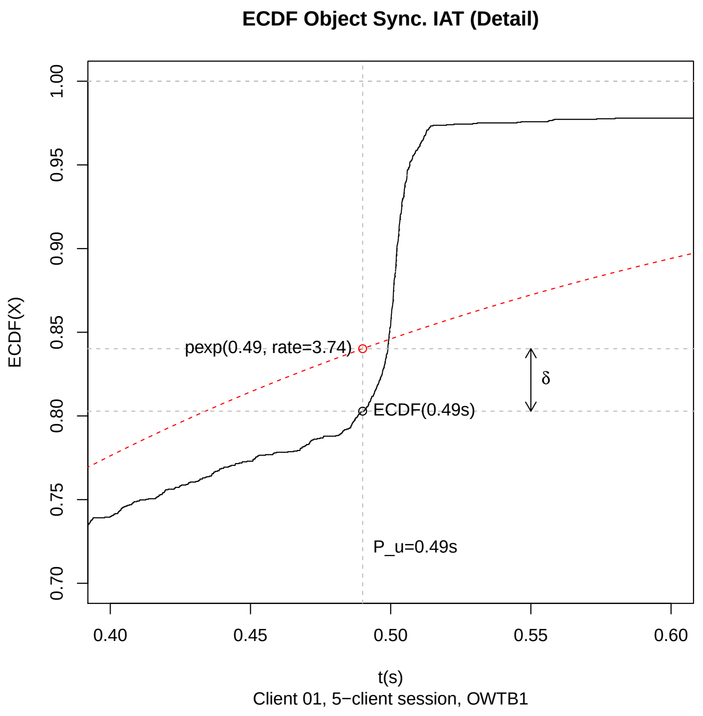


Fig. 4.8.: Detail of ECDF and fitting CDF, *Client 1*, *5-client session*, OWTB1

represented by the dashed red curve) for the point $0.49\,\mathrm{s}$, and point $ECDF(0.49s)$, part of the ECDF curve obtained for the sample values.

$$\delta = P_{FIT}(P_U, \{P_{rate\_mle}\}) - ECDF(P_U) \tag{4.1}$$

$$a_{cf} = \left(1 - \frac{\delta}{P_{FIT}(P_U, \{P_{rate\_mle}\})}\right) \tag{4.2}$$

The C_{acp} values calculated for the OSIAT models based on exponential and Weibull distributions for each client and session from OWTB1 and OWTB2 are shown in Tables D.2 and D.3 respectively, included in the Appendix D. According to these values, $C_{acp} = 0.8$ is a reasonable approximation that works well with most of the observations from the testbeds.

Fitting curve for first interval of the split distribution

The data preview from Section 4.4.1 showed that ECDF plots for OSIAT values from the activity evaluation sessions (Figure 4.4), OWTB1 multiplayer sessions (Figures D.1, D.2 and D.3), and OWTB2 multiplayer sessions (Figures D.4, D.5, D.6 and D.7), suggest that the OSIAT values follow an exponential-like distribution for values between 0 and $\approx 0.5\,\mathrm{s}$ before transitioning into a tail which weight decreases with the increase in player activity rate during the game session (Section 4.4.3). The most promising continuous probability distributions to model these values in the interval $[0, 0.5]$ are the *Exponential* and *Weibull* distributions. The present section will propose models based on each distribution.

Determining the analytical expression for the exponential section of OSIAT presents the following problem: if all the empirical data is used to calculate the exponential (or Weibull) parameters using MLE, the OSIAT values around $0.5\,\mathrm{s}$ may lead to a poorly fitting distribution that tries to both accommodate exponentially distributed values with others that do not follow this trend.

Figure 4.4 shows a preliminary fitting attempt for the IAT values from the *active/inactive* sessions from OWTB1, using a pure exponential distribution as fitting distribution. In this plot each ECDF is accompanied by two CDF plots for exponential distributions. The ones labeled as *Raw Fitting* (dashed lines) result from applying MLE to the whole set of OSIAT values from each one of the *active/inactive* sessions. The *Filtered Fitting* CDF plots (dotted lines) result from applying MLE to the set of

OSIAT values after removing the tail values, this is those values greater than $0.51\,\mathrm{s}$, as suggested in Section 4.4.3.

At these point it is useful to briefly introduce a quantitative way to determine the goodness of fit (GoF) of a given model for an empirical data set, which will numerically measure how well the proposed model describes the empirical data. The evaluation of the GoF for the proposed models is further developed and discussed in more detail in Section 4.4.5, this section will use λ^2 to determine if tail values would be included or not when using MLE to calculate the fitting curve for interval $[0, 0.5]$.

The exponential models for *Raw* and *Filtered* OSIAT values from the *inactive client* perform differently in terms of GoF, $\lambda^2_{RAW} = 0.776$ versus $\lambda^2_{FILT.} = 4.66$. On the other hand, the exponential models proposed for the values from the *active client* session show very similar values, $\lambda^2_{RAW} = 4.12$ and $\lambda^2_{FILT.} = 4.08$ respectively, which is consistent with the smaller percentage of outliers and lighter OSIAT distribution tail associated to high activity rates (Section 4.4.3).

It is noteworthy that while λ^2 provides a quantitative measure for GoF, its value does not provide an absolute measurement of how good or bad the fitness is, it just allows to compare and sort how different theoretical models approximate a given data set. While λ^2 helps to determine which model describes better than others the empirical data set, it does not provide information about *how much better* it describes it, so very different λ^2 do not necessarily imply quantitative measurement of the GoF. In the case of $\lambda^2_{RAW} = 0.776$ versus $\lambda^2_{FILT.} = 4.66$ for the proposed CDFs for the *inactive client* session, it can be concluded that filtering tail and outlier OSIAT values worsens for the modelling purposes the fit of the distribution calculated using MLE.

Removing the distribution tail values for the different testing sessions (values greater than $0.5\,\mathrm{s}$) helps to get better fitting distributions. Based on the distribution of values around $0.5\,\mathrm{s}$ shown in Figures 4.5 and 4.6, there is a low density of values greater than $0.51\,\mathrm{s}$, this can be used to delimit the distribution tail. Thus, the analytical expressions for the range $[0, t_{thres}]$ for the split distribution for OSIAT values will be obtained by applying MLE to those values from the testing sessions lower or equal than $0.51\,\mathrm{s}$.

In conclusion, the analysis suggests that OSIAT values in the interval between $0\,\mathrm{s}$ and $0.5\,\mathrm{s}$ can be potentially modelled using an exponential or a Weibull distribution. It will be required to resort to the evaluation of the statistical discrepancy between these candidate models and the experimental data set to determine which is the most suitable to describe the OSIAT for OWL.

Proposed models for Object Synchronisation Inter-Arrival Time

The result of putting together the model elements defined throughout Section 4.4.4 are Equation (4.3) and Equation (4.4), the two candidate models to describe the OSIAT values where t is the threshold for the split expression (established in $t = 0.5s$ using the *median* for OSIAT values concentrated around $0.5\,\mathrm{s}$, as described in Section 4.4.3), C_{acp} is the *activity correction parameter* which value is linked to the activity rate associated to the player within the gaming session, formulated in Section 4.4.4. λ parameter in Equation (4.3) represents the *rate* for the Exponential distribution, while λ and κ parameters in Equation (4.4) represent the *scale* and *shape* parameters respectively for Weibull distribution.

$$F(x;\lambda,C_{acp},t_{thres}) = \begin{cases} 1 \quad , & x > t_{thres} \\ C_{acp}\cdot\left(1-e^{-\lambda x}\right) \quad , & t_{thres} \geq x \geq 0 \\ 0 \quad , & x < 0 \end{cases} \tag{4.3}$$

$$F(x;\lambda,\kappa,C_{acp},t_{thres}) = \begin{cases} 1 \quad , & x > t_{thres} \\ C_{acp}\cdot\left(1-e^{-(x/\lambda)^{\kappa}}\right) \quad , & t_{thres} \geq x \geq 0 \\ 0 \quad , & x < 0 \end{cases} \tag{4.4}$$

The parameters for the exponential distribution in Equation (4.3) calculated for each client and session from OWTB1 and OWTB2 are shown in Tables D.4 and D.6 in Appendix D. Values from Table D.6 have been used to generate the Figure 4.9, a surface plot where axis X (labeled as *session*) represents the experimental sessions from OWTB2, axis Y (labeled as *client*) represents the *nth* client from a given session, and axis Z (labeled as *Exp. lambda*) is the parameter value obtained by MLE. Thus, a point (x, y, z) represents the λ exponential parameter of value z calculated applying MLE to the OSIAT values from the y^{th} client in the x^th experimental session performed in OWTB2. The surface shows the MLE values are around 0.47.

The parameters for the Weibull distribution in Equation (4.4) calculated using for each client and session from OWTB1 and OWTB2 are shown in Tables D.5 and D.7 in Appendix D. Values from this last table have been used to generate the surface plots in Figure 4.10. As in the case of Figure 4.9, each point (x, y, z) represents the parameter calculated using MLE with value z for the y^{th} client in the x^{th} session performed in OWTB2. These surfaces show that the *shape* and *scale* parameters for Weibull distribution take values around 0.51 and 0.14 respectively.

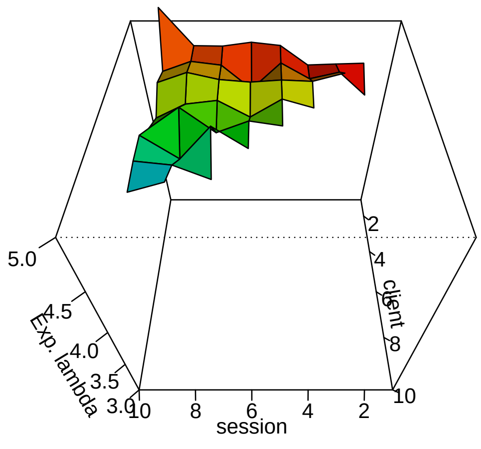
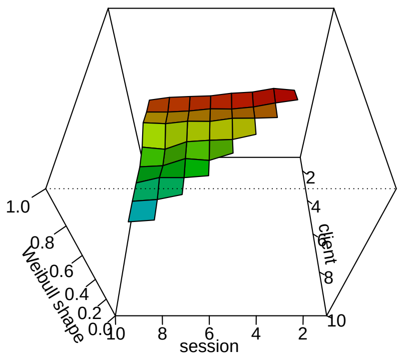


Fig. 4.9.: Exponential λ rate values for OWTB2 clients and sessions



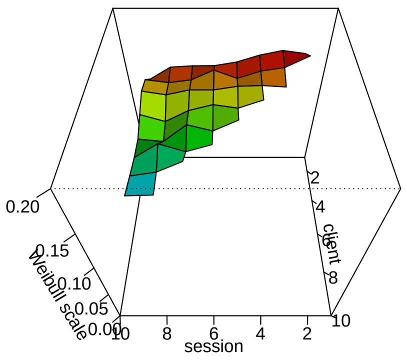


Fig. 4.10.: Weibull λ scale and κ shape values for OWTB2 clients and sessions

Table 4.3 contains a summary of the values from tables D.4, D.6, D.5 and D.5.

Tab. 4.3.: Summary of MLE parameters for OWTB1 and OWTB2

Distribution	Parameters	OWTB1		OWTB2	
		μ	σ^2	μ	σ^2
Exponential	λ rate	3.63	0.74	4.76	0.10
Weibull	λ scale	0.26	0.05	0.14	0.01
	κ shape	0.87	0.11	0.51	0.03

The λ parameter values obtained by MLE for the exponential-based model are quite similar between the OSIAT data flows generated by each one of the clients engaged in the same gaming session. These same λ values seem to be correlated to the number of players within a given session. The same trends are observed for parameters λ and κ for Weibull-based model. This suggests that the number of simultaneous players engaged in the same OWL gaming session has an impact on the ratio of object synchronisation packets that are received from each client on the OWL server side.

4.4.5 Statistical discrepancy

The statistical discrepancy or GoF for the OSIAT is evaluated using the following methods suggested by the analysis pipeline (must be noted that Pearson correlation coefficient (PCC) has been also included):

- Q-Q plots to provide an intuitive and qualitative preview of GoF.
- Pearson Correlation Coefficient to measure the similarity between values generated using the ECDF and the proposed OSIAT models.
- $\hat{\lambda}^2$ to quantitatively compare the GoF provided by the Exponential and Weibull-based OSIAT models formulated in Section 4.4.4.

Q-Q plots

Q-Q plots (Section 3.8.1) make possible to qualitatively compare quantiles from two distributions, in the current case, the quantiles derived from the ECDF for the OSIAT versus the quantiles for the models Equation (4.3) and Equation (4.4) proposed in Section 4.4.4. These models only describe the exponential-like interval of OSIAT values in interval $[0, t_{thres}]$ and the probability saturation point at t_{thres}, leaving out the tail of OSIAT values. Thus, it is expected that the Q-Q plots will show a poor correspondence between empirical and model quantiles (points largely deviating from the plot diagonal). In the present study Q-Q plots are used to provide a qualitative measure of GoF for the OSIAT quantiles in the interval $[0, t_{thres}]$.

Due to space limitations, the present section only includes the Q-Q plots for the *3-client* sessions from OWTB1 and OWTB2. A greater selection of OSIAT can be found in Appendix D, Section D.6. Figure 4.11 suggests a good fit between the empirical quantiles and those calculated for Equations (4.3) and (4.4) for OWTB1. The plot includes the values used for the model parameters.

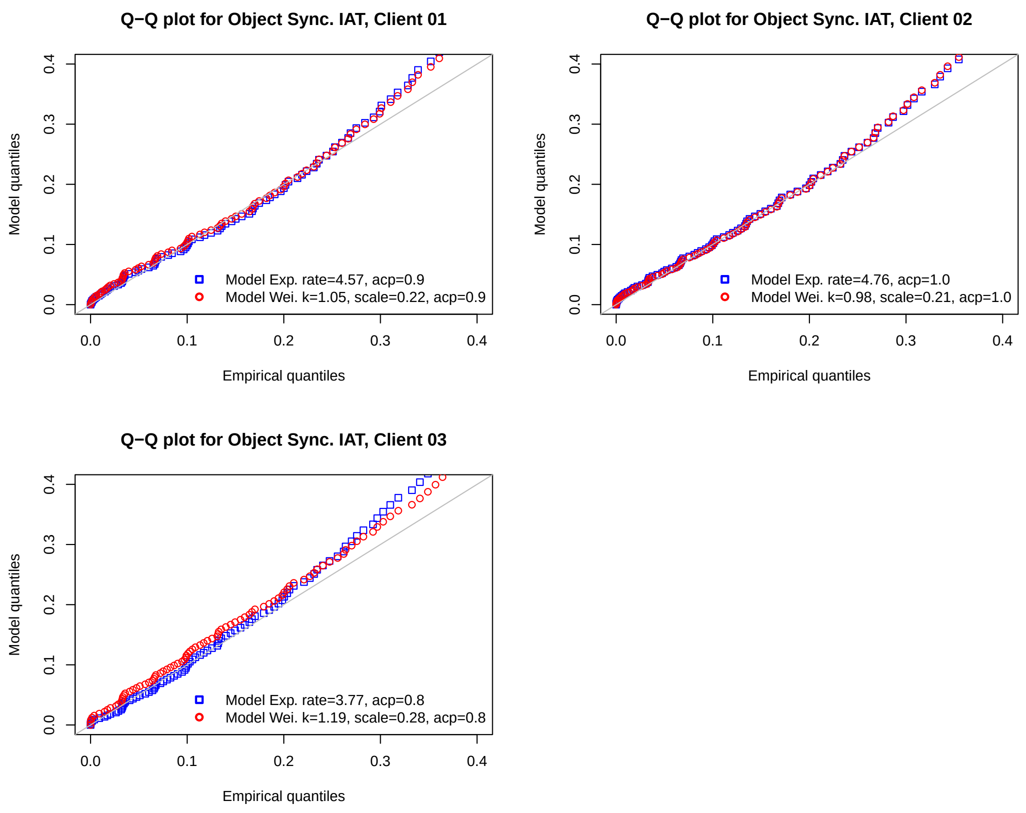


Fig. 4.11.: Q-Q for OSIAT models, *t3-player* session, OWTB1

Figure 4.11 shows a slightly worse fit for the OWTB2 sessions. It can be observed that both the Weibull and Exponential-based models make the points in the Q-Q plot fall above the diagonal, which means they overestimate the weight of the interval $[0, t_{thres}]$ in the overall model. This deviation from the diagonal stays constant for many of the quantiles, which suggests the possibility to improve the fitting by correcting such an offset in the proposed OSIAT models.

On the other hand, Figure 4.12 shows the Q-Q plots for the 3-client session from OWTB2. These plots show points run parallel to the bisector of the first quadrant. The offset making the model quantiles (axis Y) being above the bisector suggests the model slightly overestimates the values for the cumulative probability for empirical results from OWTB2.

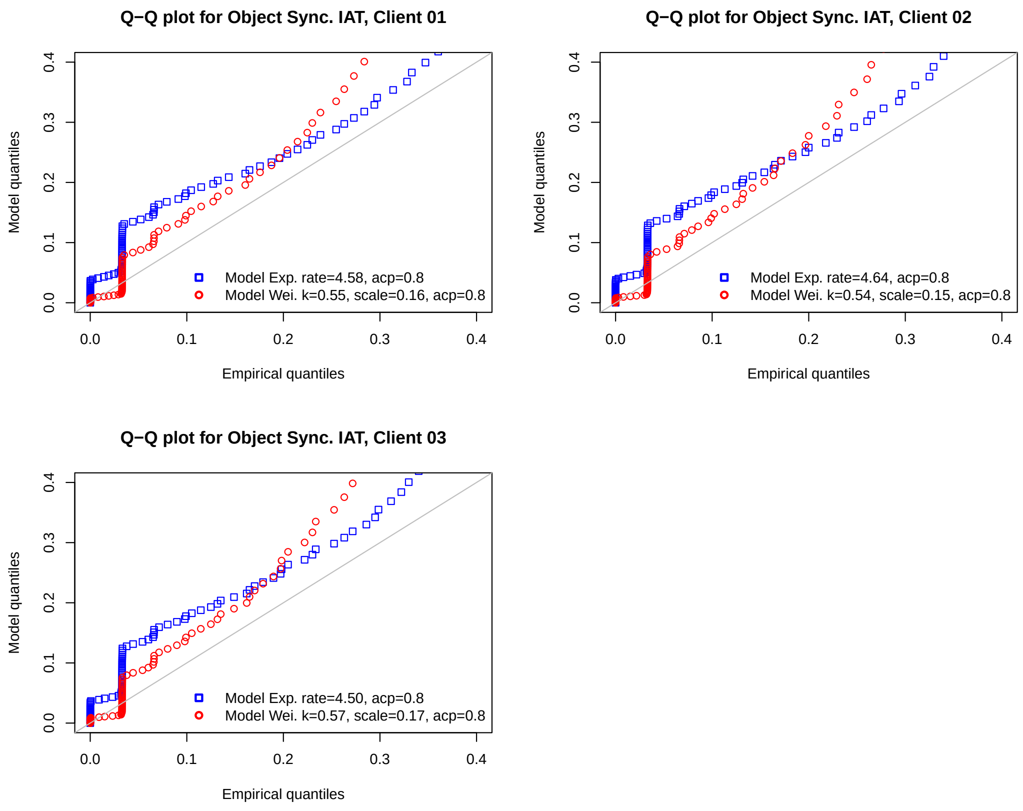


Fig. 4.12.: Q-Q for OSIAT models, *3-player* session, OWTB2

While providing an intuitive estimation of the GoF, the Q-Q plots do not allow to easily determine which of the proposed OSIAT models (exponential vs. Weibull) provides a better fit for the empirical data, being advisable to resort to quantitative methods to be able to compare the level of GoF provide by each candidate model.

Pearson Correlation Coefficient

The PCC or *Pearson's r* measures the linear correlation between two variables x and y, where r measures the trend relative magnitude of the fit [Fis15], [DGK75]. A r value equal to 1 indicates that the equation describes the relationship between x and y perfectly, while a r equal to 0 implies the lack of correlation between empirical data and the model. The Pearson's r is accompanied by r^2, which indicates the proportion of the total variance that is explained by the prediction. While PCC is not a robust statistic due to its sensibility to outliers, it is used in the present study to provide a quick and simple estimation of GoF by measuring the correlation between empirical values and those derived from the proposed theoretical distributions.

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \quad (4.5)$$

Two PCC are calculated for each experimental session, derived from evaluating the ECDF with the exponential and Weibull-based models. The procedure involves calculating the probabilities for the sequence $(s_i)_{i=0}^{50} = \frac{i}{100}$ using $ECDF(s_i)$, and the CDF for the Exponential and Weibull-based OSIAT models ($F_{exp}(s_i|\lambda, C_{acp,exp})$ and $F_{wei}(s_i|\kappa, \lambda, C_{acp,wei})$ respectively).

The choice of the sequence s_i aims to provide a rough evaluation of GoF of the OSIAT models ignoring tail values, which are out of the scope of the models and would also penalize the results for PCC.

Tables D.8 and D.9 in Appendix D show the resulting PCC values, where r_{exp} results from applying Pearson's r to $ECDF(s_i)$ and $F_{exp}(s_i|\kappa, \lambda, C_{acp,exp})$, while r_{wei} results from applying PCC to $ECDF(s_i)$ and $F_{wei}(s_i|\kappa, \lambda, C_{acp,wei})$. Tables of r results show both Exponential and Weibull-based models provide very similar probability values to those from ECDF for values in $[0s, 0.5\,\mathrm{s}]$ (r values close to 1) but probabilities derived from the Weibull-based model $F_{wei}(s_i|\kappa, \lambda, C_{acp,wei})$ have a higher degree of correlation with the empirical data sets.

Lambda square λ^2 and Hat lambda square $\hat{\lambda}^2$

The GoF metric $\hat{\lambda}^2$ described in Section 3.8.2 is a refinement of λ^2 aimed to provide a higher degree of robustness by reducing the sensitivity to tails, extreme values and division-by-zero situations.

Due to space limitations, the present section only includes the $\hat{\lambda}^2$ values obtained for the empirical OSIAT values versus the Exponential (Equation (4.3)) and Weibull-based (Equation (4.4)) OSIAT values for each client involved in the 2, 3 and 5-client sessions for both OWTB1 (Table 4.4) and OWTB2 (Table 4.5) respectively. The complete $\hat{\lambda}^2$ results can be found in Tables D.10 and D.11, Appendix D.8.

$\hat{\lambda}^2$ values for OWTB1 calculated for Exponential and Weibull-based OSIAT models in Table 4.4 are very similar for each client and session, most of the time showing differences in the range of tenths. The two models alternate providing the best fit for the different samples.

Tab. 4.4.: $\hat{\lambda}^2$ for exponential and Weibull-based models, OWTB1

Session	Client	$\hat{\lambda}^2_{exp}$	$\hat{\lambda}^2_{wei}$	$\sigma(\hat{\lambda}^2_{exp})$	$\sigma(\hat{\lambda}^2_{wei})$
2-client	1	14.33	14.26	837.45	824.49
	2	18.30	18.33	1347.37	1353.20
3-client	1	12.26	12.21	608.85	601.73
	2	4.37	4.38	86.77	88.16
	3	6.40	6.22	181.17	159.32
5-client	1	9.26	9.08	372.39	345.91
	2	7.74	7.78	265.54	271.29
	3	4.46	4.14	94.35	70.77
	4	23.14	24.70	2147.04	2778.04
	5	7.69	7.23	298.48	215.11

$\hat{\lambda}^2$ values for OWTB2 calculated for Exponential and Weibull-based OSIAT models in Table 4.5 show a similar trend to those from Table 4.4 but in this case, and others from OWTB2, the Weibull-based model provides a better fitting that the Exponential model.

Tab. 4.5.: $\hat{\lambda}^2$ for exponential and Weibull-based models, OWTB2

Session	Client	$\hat{\lambda}^2_{exp}$	$\hat{\lambda}^2_{wei}$	$\sigma(\hat{\lambda}^2_{exp})$	$\sigma(\hat{\lambda}^2_{wei})$
2-client	1	4.60	4.12	277.98	210.49
	2	3.26	2.60	324.23	200.11
3-client	1	2.81	2.09	148.40	75.77
	2	1.97	1.52	62.60	55.19
	3	1.60	1.17	23.56	12.98
5-client	1	3.03	2.63	52.06	44.78
	2	1.85	1.20	265.23	130.62
	3	2.32	1.82	65.93	42.04
	4	5.83	5.56	149.43	138.94
	5	2.66	2.09	128.92	83.14

Looking at the values of $\hat{\lambda}^2$ obtained for OSIAT values from OWTB2, it would be possible to favour the Weibull-based OSIAT model formulated in Equation (4.4) over its exponential counterpart in Equation (4.3). Nevertheless, the suitability of one of other candidate will be conditioned by the context in which the candidate models should be used and its specific goals.

4.5 Object synchronisation packet size

The term object synchronisation packet size (OSPS) defined in the present work refers to the size, in bytes, of the object synchronization packets arriving to a given host, in this case, the server. The OSPS is one of the two variables used in the microscale modelling approach for modelling NVE network traffic. The following sections describe how the OSPS has been measured and analysed to produce an statistical model. Regarding packet size measurement, only the TCP packet payload has been considered, the overhead due to the different transport protocols that encapsulate the payload, such as TCP, IP or Ethernet, has been omitted.

4.5.1 Data preview

The OSPS values studied in the present section include the size of the TCP/IP headers, this is around $54\,\mathrm{B}$ that encapsulate the payload of the Object Synchronisation traffic itself.

The barplot for the combined OSPS values generated during the different gaming sessions on OWTB1, Figure 4.13, shows that the most frequent OSPS value by far is $293\,\mathrm{B}$, comprising up to $96.6\,\%$ of all the packet sizes registered in OWTB1.

On the other hand, barplot in Figure 4.14 for all the OSPS values from OWTB2 shows only two values for all the sessions ($293\,\mathrm{B}$ and $304\,\mathrm{B}$). This reduction in the amount of possible values for OSPS and distribution can be explained by the technical characteristics of OWTB2, where the avatars were controlled by scripts, programmed to keep high rates of avatar activity within the gaming sessions. This high rate of events can explain a smaller and fixed-size Object Synchronization packets containing the minimum payloads to keep the shared simulation properly synchronised among all the players.

The data preview for the OSPS shows a discrete distribution of the values, suggesting a discrete model can capture the nature of the random variable.

4.5.2 Autocorrelation

The ACF plot in Figure 4.15 does not suggest autocorrelation in the distribution of OSPS values from 2-client session in OWTB1.

The ACF plot in Figure 4.16, OSPS values for 2-client session in OWTB2, on the other hand shows some signs of autocorrelation and inverse autocorrelation after certain number of lags. This is a side effect of the automation of the avatar control and how it has pushed to the maximum the rate of the Object Synchronisation traffic, which translated into burst of update packets of very similar packet size.

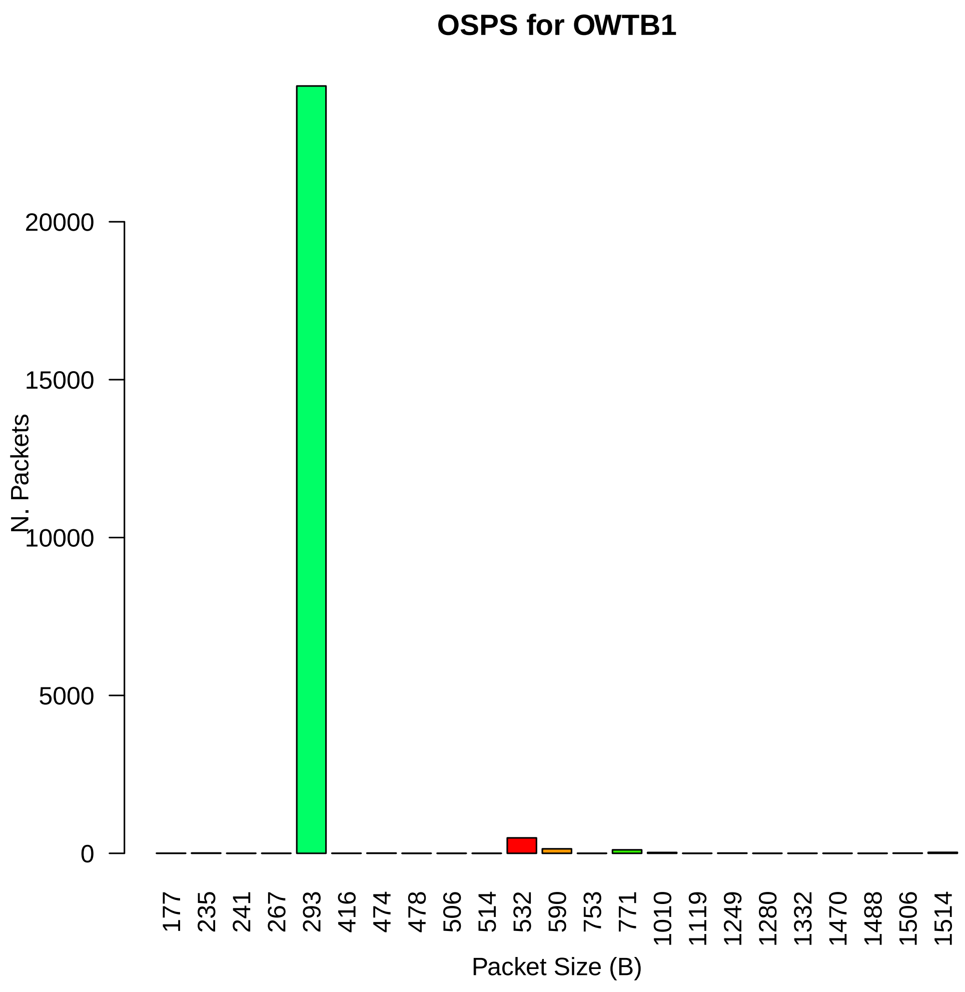


Fig. 4.13.: OSPS values for OWTB1

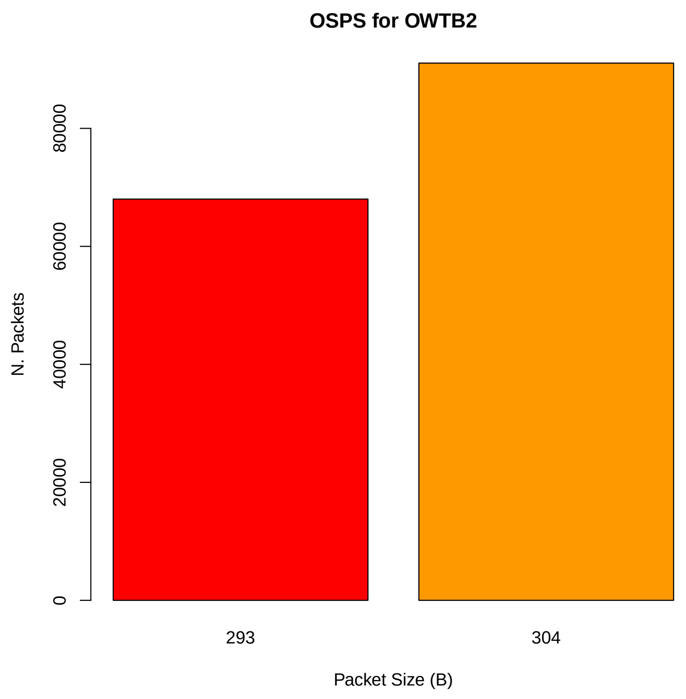


Fig. 4.14.: OSPS values for OWTB2

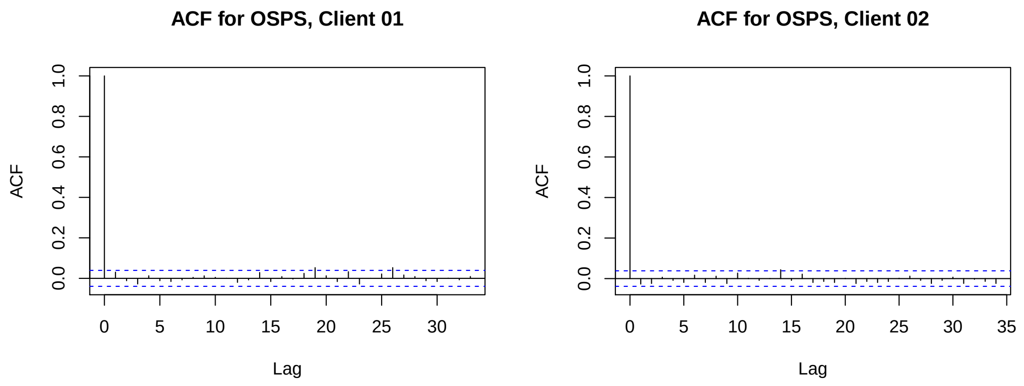


Fig. 4.15.: ACF for OSPS values, *2-player* session, OWTB1

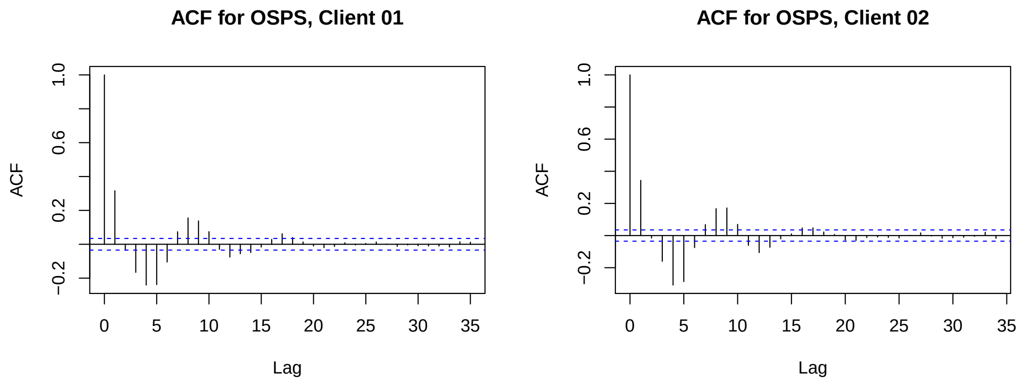


Fig. 4.16.: ACF for OSPS values, *2-player* session, OWTB2

Section 4.5.1 suggests a simple distribution of OSPS values, so even if there are some signs of autocorrelation in the empirical data, this aspect will be left out the scope of the modelling process in this work.

4.5.3 Data Modelling

Data preview for OSPS in Section 4.5.1 suggests that it can be modeled with a simple discrete probability distribution tailored for the empirical results obtained from OWTB1 and OWTB2.

The relative frequencies for OSPS obtained from OWTB1 are shown in Table 4.6. Columns show the four most frequent packet size values for each session plus a last row with the frequencies for the aggregation of all the packets generated on the testbed.

Tab. 4.6.: Relative frequency for OSPS, OWTB1

Session	N. Packets	293 B	532 B	590 B	771 B	Other
2-client	5099	0.968	0.021	0.000	0.008	0.003
3-client	7768	0.965	0.021	0.005	0.005	0.004
5-client	12286	0.966	0.019	0.008	0.002	0.005
Total	25153	0.966	0.019	0.006	0.004	0.004

Values from Table 4.6 allow to define the simple ad hoc model for OSPS in Equation (4.6).

$$P(x) \quad = \quad \begin{cases} 0.97 & , \quad x = 293 \quad bytes \\ 0.02 & , \quad x = 532 \quad bytes \\ 0.01 & , \quad x = 590 \quad bytes \end{cases} \tag{4.6}$$

The relative frequencies for OSPS measured from OWTB2 are shown in Table 4.7. Columns show the four most frequent packet size values for each session plus a last row with the frequencies for the aggregation of all the packets generated on the testbed.

Values from Table 4.7 allow to define the simple ad-hoc model for OSPS in Equation (4.7).

Tab. 4.7.: Relative frequency for OSPS, OWTB2

Session	N. Packets	293 B	304 B
01-client	2809	0.41	0.59
02-client	6358	0.42	0.58
03-client	9074	0.44	0.56
04-client	12087	0.43	0.57
05-client	14524	0.43	0.57
06-client	17776	0.43	0.57
07-client	19977	0.43	0.57
08-client	22636	0.43	0.57
09-client	26101	0.42	0.58
10-client	27736	0.43	0.57
Total	159078	0.43	0.57

$$P(x) = \begin{cases} 0.43 & , \quad x = 293 \quad bytes \\ 0.57 & , \quad x = 304 \quad bytes \end{cases} \tag{4.7}$$

Choosing one model over the other will depend on the specific setup that is going to be modelled. OWTB1 is a more heterogeneous testbed, where avatars where operated by human players, while OWTB2 is an homogeneous testbed where avatars performed a high rate of activity and interaction within the sessions due to the automatic control performed with scripts, providing a relatively *synthetic* setup which on the other hand allows to evaluate the features and performance of the system.

Due to the simplicity of the OSPS models in Equations (4.6) and (4.7), the study of GoF has not been included.

4.6 Summary

The present chapter details the analysis details and results from applying the pipeline proposed in Chapter 3 to the experimental network data generated for the NVE software OWL in the testbeds from Chapter 2 (OWTB1 and OWTB2). The goal of this exercise is to determine the suitability of the pipeline to model the network traffic generated by a system, OWL, fitting into the definition of NVE provided in Section 2.1. Thus, the analysis pipeline has been used to model the random variables OSIAT and OSPS which describe the OWL network traffic at microscale level.

This chapter provided a technical description of the *object synchronisation traffic* generated by OWL, where the packets belonging to these data flows will be filtered and object of study using the analysis pipeline. It was also provided a naming convention and definition for the random variables associated to the *object synchronisation traffic*, OSIAT and OSPS, which describe the behaviour of the network traffic at microscale level.

It is also included the detailed study and modelling of the OSIAT variable for the OWL *object synchronisation traffic* from the testing testbeds, performed using the analysis pipeline proposed in the previous chapter. This study included:

- The data preview of the OSIAT revealed an exponential-like distribution of values and a probability saturation around value $0.5\,\mathrm{s}$.
- Autocorrelation study showed OSIAT values are not autocorrelated and match the requirements to be modelled as a random variable.
- The study of the user activity within the gaming sessions revealed that there is a correlation between the activity and the size of the tail of the distribution of OSIAT, the higher the activity, the smaller the distribution tail.
- The data modelling step proposed a split distribution to model the OSIAT, defining all the parameters that determine the behaviour of the model and providing its analytical expression. Because of the exponential-like look of the data preview, two variations of the OSIAT model were proposed: one based on the exponential distribution for values between 0 and $0.5\,\mathrm{s}$ (Equation4.3) and another based on the Weibull distribution for the values in this same range (Equation 4.4). The models were accompanied by the parameters for the exponential and Weibull components, calculated by applying the MLE method to the experimental data obtained from the testbeds (Table 4.3).
- The measurement of the statistical discrepancy was performed using several methods: Q-Q plot to have an intuitive estimation of the GoF, PCC as a simple numeric metric for fitness, and λ^2 / $\hat{\lambda}^2$ a more robust metric for statistical discrepancy which allows comparing the degree of fitting between the experimental data and the proposed models. The Q-Q plots did not show a clear advantage to any of the two proposed models. According to the $\hat{\lambda}^2$ results, the GoF for the Weibull-based OSIAT was better than that for the exponential-based model (Tables 4.4 and 4.5).

The study and modelling of the microscale variable OSPS, performed using the analysis pipeline included the following elements:

- The data preview showed that OSPS values are a small set of discrete values, which made evident the convenience of modelling the variable with a discrete probability model.

- The autocorrelation study showed little evidence of self-similarity, so OSPS can be considered as a random variable and modelled as such. It was also observed that the number of number of discrete values for OSPS varied from one testbed to another, a side effect of the different rates of user activity and network traffic present in each one.
- The data modelling step provided a simplified discrete model for OSPS described in Equation 4.7.

5 Simulation based on NVE models

Maybe the only significant difference between a really smart simulation and a human being was the noise they made when you punched them.

Terry Pratchett, *The Long Earth*

This chapter describes the design, implementation and execution of simulations based on the object synchronisation inter-arrival time (OSIAT) models proposed in Chapter 4, Section 4.4.4 for Open Wonderland (OWL) network traffic. These models were generated using the analysis pipeline for networked virtual environment (NVE) network traffic defined in Chapter 3. Results from these simulations will help to validate the models implemented by the simulation and also the pipeline that generated those models. Moreover, these validated models will be ready for further use in research of performance evaluation of NVE systems in the lane of OWL.

Simulation is a key tool in networking research due to its inherent advantages over testing with real hardware [RPW09]. Matters such as budget and cost of time make simulation attractive for research, while flexibility, scalability and virtually no cost expansion are also valuable advantages over other testing methods [OB10]. Network simulators allow us to implement and study different network entities in a simulated environment, providing a high degree of flexibility to test new protocols, technologies, conceptual models and topologies.

5.1 Simulation framework: ns-3

The implementation of the models for OSIAT and object synchronisation packet size (OSPS) proposed in Chapter 4 (Sections 4.4.4 and 4.5.3 respectively) is based on the simulation tool and framework *ns-3* [Pro19]. The version used in the present work is ns-3.29, last stable release at the time of writing. The choice of ns-3 is derived from the author's previous work in [Fon+10] and [Fon+11a]. These works approach the comparison of *ns-2* and *ns-3*, two of the most popular network simulation frameworks at the time of writing them, from the perspective of software engineering by using complexity and code quality metrics.

The chosen network simulation framework, *ns-3*, provides a modular and well tested simulation framework, it is an open source community-driven project and provides a well defined building process, comprehensive documentation and the possibility of coding in C++ or Python [Fon+10], [Fon+11a]. Ns-3 is devised to be executed in GNU/Linux and Unix-like environments, being also possible to build and run it in Windows systems using Cygwin. Ns-3 was conceived as ns-2's [Proa] [LA03] successor: its developers have tried to solve or mitigate many of ns-2's well-known drawbacks as well as apply new concepts, such as validation and software engineering techniques, to produce a more reliable simulation tool to support academic and industrial research [Hen14].

Ns-3 is conceived to be distributed in source code form though some GNU/Linux distributions such as Debian or Arch provide pre-compiled packages. Due to users have access to source code, they can modify and extend their features and optimize their binaries as needed. Source code can be fetched from the Git [Prob] repository maintained by the ns-3 project. This repository contains all the official releases, and publicly available development branches.

The programming language used for the ns-3 core and model components is C++, while simulations that put to work these components can be written in C++ or Python, an interpreted language which presents some advantages about development speed and code readability.

Regarding the architecture of the framework, ns-3 emphasizes source code hierarchical structuring by defining several basic network entities present in every single simulation. Specific models are a refinement of these generic entities [Pro19].

- *Node*, represents the basic computing device. It acts as container of the elements of the network stack and interacts with other nodes by the communication channel. Nodes require net devices in order to be able to use the physical channel.
- *Application*, sets up on top of the network stack within the node, playing the role of packet consumer or packet generator.
- *Channel*, provides the media to interconnect nodes and to allow data traffic exchange.
- *Net Device*, abstracts the network hardware that makes communication possible through a channel.

- *Topology Helper*, auxiliary class that makes the generation of complex topologies easier by automating the creation of network elements, their configuration and interconnection.
- *simulation*, a program that defines, configures and launches the simulation experiment based on the models and helpers.

Ns-3 gathers all source code files of both simulator core and models in the `ns-3/src` folder, separating them from the rest of auxiliary tools, building scripts, documentation and examples. The `ns-3/src` folder is also subdivided, each subfolder containing the files that implement the different models shipped with ns-3 by default, plus the simulator core components. On the other hand, simulation are placed in the `ns-3/scratch` folder.

5.2 Simulation implementation

Ns-3 code is divided into models, helpers and simulations. The models are classes that contain the functionality of the modelled system, such as network nodes, data packets, protocol stacks or network devices. On the other hand helpers are auxiliary classes that allow to create and configure several objects of the same class in an easy and concise way, such as defining several interconnected nodes, initialising a large amounts of network devices with consecutive IP addresses, and so on. Finally, a simulation is a piece of code that defines, configures and launches the simulation experiment based on the models and helpers provided by ns-3.

While ns-3 simulations can be written in C++ or Python, the former is the language for model and core components. The ns-3 models, helpers and simulation coded and used in the context of the present work have been written in C++. Many of the classes mentioned below contain the prefix *ns3::*, which is the C++ namespace for all ns-3 classes.

5.2.1 Implementation and code structure

Figure 5.1 contains a simplified diagram of the different entities that are part of the OSIAT simulation. The simulation rely on OWL client and server, ns-3 nodes each one sporting a different collection of models to simulate the object synchronisation exchange of traffic. These nodes relies on the ns-3 TCP/IP stack and carrier sense multiple access (CSMA) models to exchange information between them. The dashed rectangles within the figure represent the helpers that assist in the creation and configuration of the different kind of nodes in the simulation.

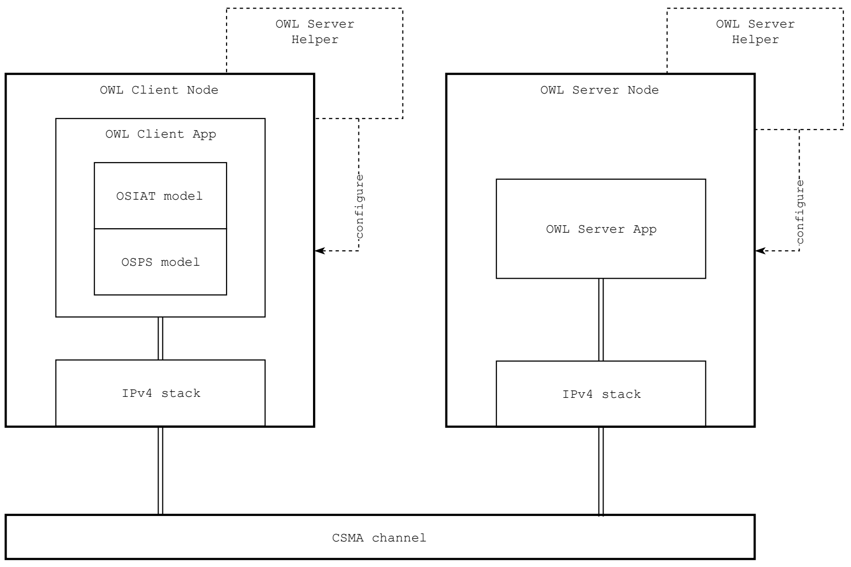


Fig. 5.1.: Structure of the ns-3 simulation for OSIAT

Below there is an enumeration of the main object oriented (OO) classes that constitute the building blocks for the ns-3 simulation based on the OSIAT and OSPS models from Chapter 4. Each class includes a brief description and relationships with other classes.

ns3::Node acts as a container entity that represents a network node. It contains objects that simulate network devices, protocol stacks or applications that generate or consume network packets. Auxiliary classes such as *ns3::NodeContainer* help to manage a large number of *ns3::Node* objects.

ns3::Application represents elements that generate and consume network traffic following the logic coded by the programmer, e.g. a Telnet server.

WonderMovClient extends *ns3::Application* class, it uses the models for OSIAT and OSPS proposed in Sections 4.4.4 and 4.5.3 to generate traffic the way OWL clients do. It relies on ns-3 models for the TCP/IP stack, such as *ns3::Socket* and *Ipv4Address*. This application requires to be connected to an object of class *WonderMovServer* which plays the role of OWL server. Figure 5.2 shows the relationship of this class with the rest of classes in the simulation.

The operation of the application is shown in the sequence diagram in Figure 5.3. Once the client is associated to the server object, it starts a loop that determines the size of the packet to be sent using the OSPS model, then proceeds to send the packet. The server receives this packet, processes it and proceeds to send an ACK message to the client. The client uses a OSIAT model to determine when the next packet will be sent to the server, starting a new iteration of the loop.

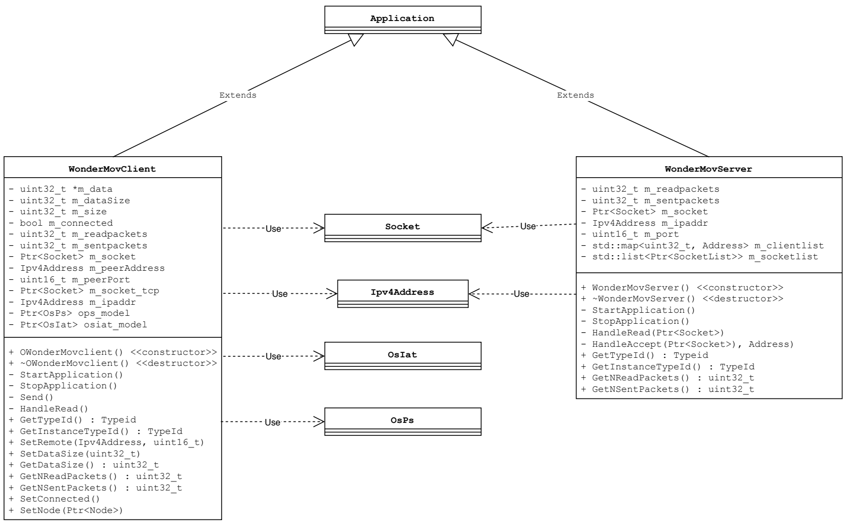


Fig. 5.2.: Class diagram for *WonderMovClient* and *WonderMovServer*

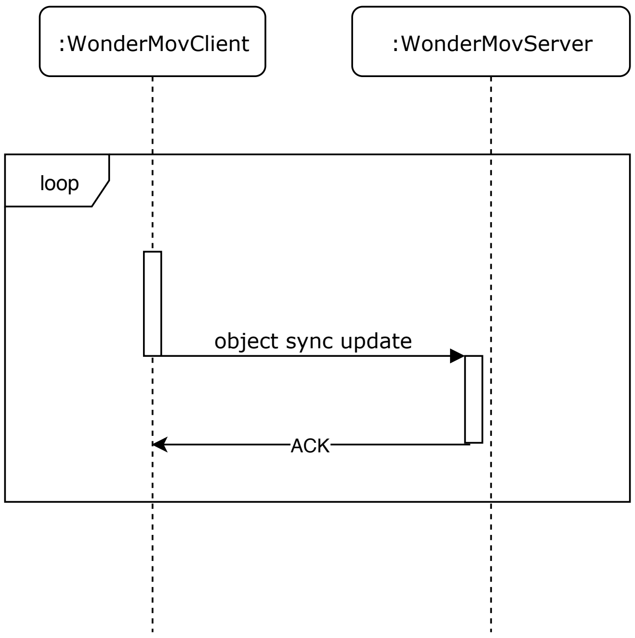


Fig. 5.3.: Sequence diagram for OWL traffic models

WonderMovServer extends *ns3::Application* class. Objects of this class are associated to *WonderMovClient* objects which send them object synchronisation packets. The class models the very basic functionality of the OWL server, only reading packets and proceeding to acknowledge them. The class is also capable of propagating the object synchronization updates to the rest of the client objects associated to the server. Figures 5.2 and 5.3 show its class and sequence diagram respectively. It must be noted that the object synchronisation traffic sent by the OWL server is mostly reactive, without signs of periodicity nor bursty behaviour. This detail has been taking into account when implementing the *WonderMovServer* class.

WonderMovClientHelper and WonderMovServerHelper helper classes that provide auxiliary methods to help with the configuration of several nodes running the *WonderMovClient* and *WonderMovServer* applications described above. The helpers take care of installing the corresponding application in each one of the provided nodes. Figure 5.4 shows the class relationships and a summary of the methods and attributes for these classes.

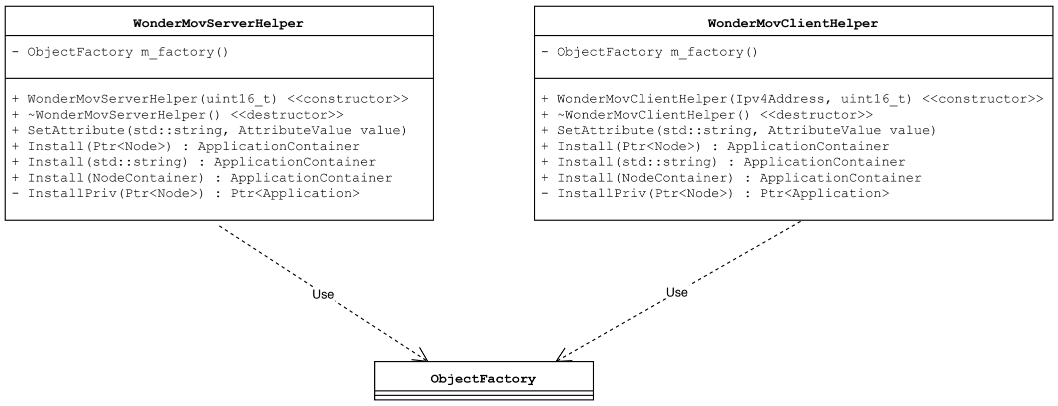

WonderMovServerHelper
- ObjectFactory m_factory()
+ WonderMovServerHelper(uint16_t) <<constructor>>
+ ~WonderMovServerHelper() <<destructor>>
+ SetAttribute(std::string, AttributeValue value)
+ Install(Ptr<Node>) : ApplicationContainer
+ Install(std::string) : ApplicationContainer
+ Install(NodeContainer) : ApplicationContainer
- InstallPriv(Ptr<Node>) : Ptr<Application>
WonderMovClientHelper
- ObjectFactory m_factory()
+ WonderMovClientHelper(Ipv4Address, uint16_t) <<constructor>>
+ ~WonderMovClientHelper() <<destructor>>
+ SetAttribute(std::string, AttributeValue value)
+ Install(Ptr<Node>) : ApplicationContainer
+ Install(std::string) : ApplicationContainer
+ Install(NodeContainer) : ApplicationContainer
- InstallPriv(Ptr<Node>) : Ptr<Application>
Use
Use
ObjectFactory

Fig. 5.4.: Class diagram for *WonderMovClientHelper* and *WonderMovServerHelper*

OsIat, OsIatExp and OsIatWei *OsIat* is an generic class to describe the common functions to all the OSIAT models. Specific ones such as *OsIatExp* (based on the Exponential distribution) or *OsIatWei* (based on the Weibull distribution) extend this class. The *WonderMovClient* application is implemented to work with any class that implements the methods defined in this class, making it easy to incorporate further OSIAT models to the simulation. *OsIatExp* and *OsIatWei* classes rely on several *ns3::RandomVariableStream* derived classes such as *ns3::ExponentialVariable*, *ns3::UniformVariable* or *ns3::WeibullVariable* to implement the models described in Section 4.4.4. Methods, attributes and relationships between classes are shown in the class diagrams in Figures 5.5 and 5.6.

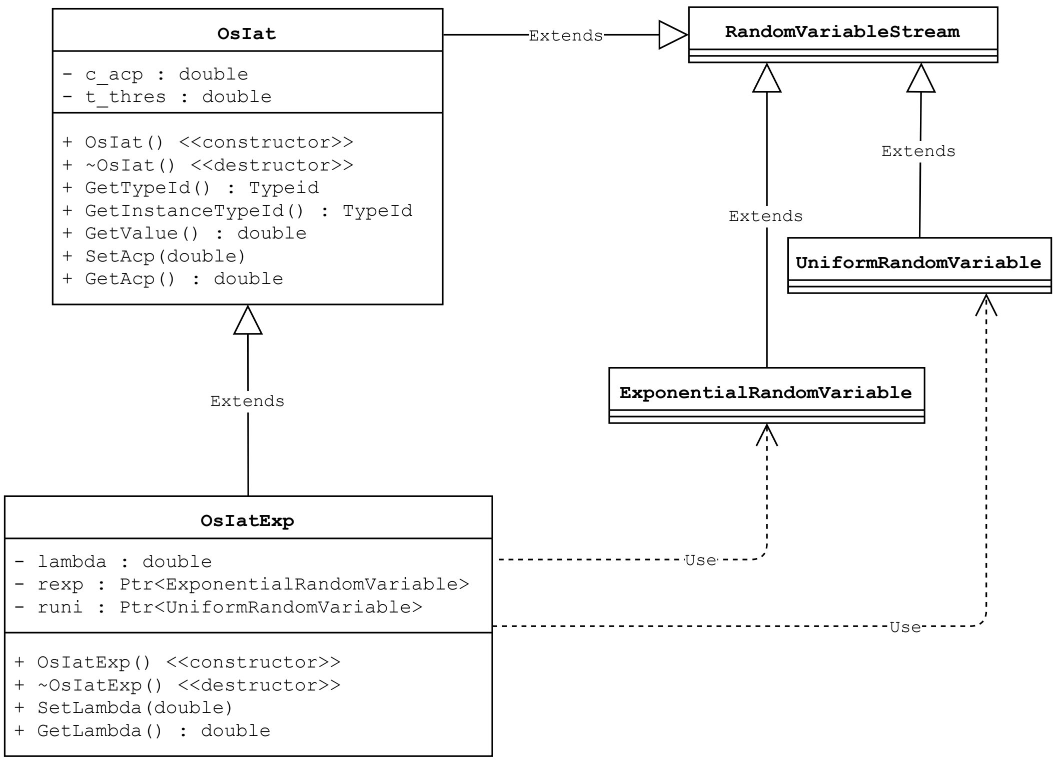


Fig. 5.5.: Class diagram for exponential-based OSIAT model

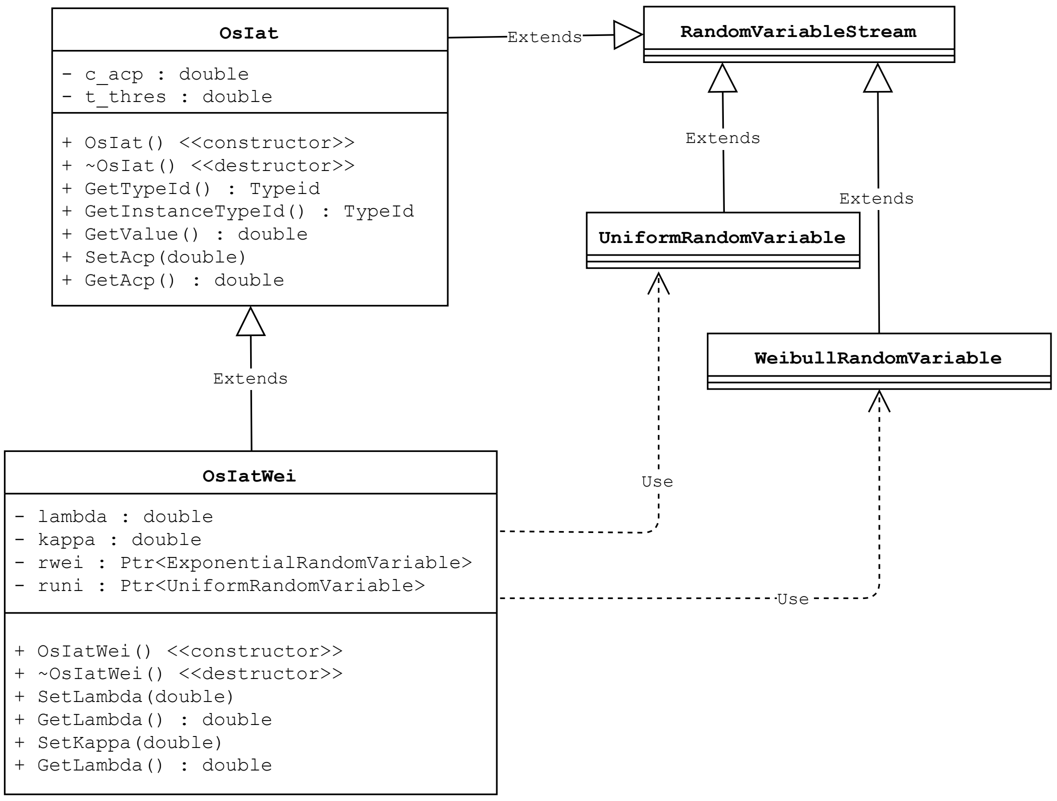


Fig. 5.6.: Class diagram for Weibull-based OSIAT model

OsPs implements the OSPS model proposed in Section 4.5.3. It relies on the class *ns3::UniformVariable* provided by ns-3. Methods, attributes and relationships with other classes are show in the class diagram in Figure 5.7

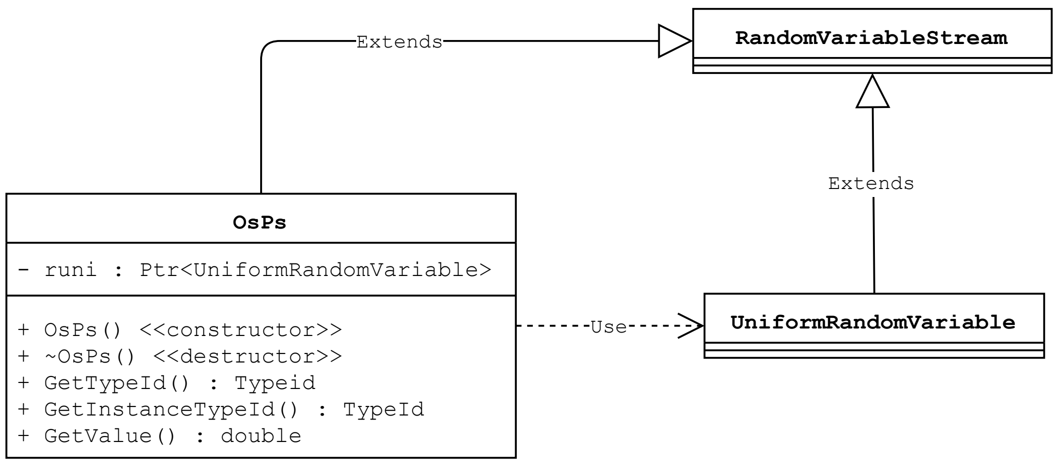


Fig. 5.7.: Class diagram for OSPS model

5.2.2 Simulation executable

The simulation executable uses all the models and auxiliary classes described in previous sections to define a simulation scenario governed by a set of parameters such as number of nodes, speed of the channel, or packet model parameters. This element is the entry point for the simulation parameters, allowing the user execute the simulation via command-line interface (CLI) following their specifications.

This simulation comprises a CSMA bus topology operating at 100 Mbps which interconnects an arbitrary number of OWL clients to a single server. This topology is equivalent to the original 100 Mbps Ethernet network used in the testing sessions. All the simulation nodes share the same CSMA channel and are equipped with CSMA network devices. Each node has also an IPv4 stack and associated IP address. All these models are part of the ns-3 core and are well tested and documented [Pro19].

The *Application* child classes are installed in all the nodes in the simulation. Each client node has a *WonderMovClient* object associated while the server node has a *OWLMovServer* associated. The simulation also makes use of several helpers part of the ns-3 core, such as *ns3::InternetStackHelper*, *ns3::Ipv4AddressHelper*, *ns3::CsmaHelper* or *ns3::Ipv4GlobalRoutingHelper* to automate the network configuration of an arbitrary number of nodes in an easy way.

The simulation accepts several arguments to shape and configure the simulation scenarios:

`-verbose=[true|false]` boolean value, enables/disables verbose log output. Default value is *true*.

`-log-pcap=[true|false]` boolean value, saves the network traffic generated by the simulator in pcap format. Default value is *true*.

`-log-ascii=[true|false]` boolean value, saves a summary of the network traffic generated by the simulator in ASCII format. Default value is *false*.

`-log-path='log-folder'` string value, specifies the path where logs, both pcap and/or ASCII format, will be stored. The path *log-folder* has to exist at the time of running the simulation. Default path is the *$HOME* folder for the user running the simulation.

`-log-name='trace-name'` string value, specifies the root name for the log files, both pcap and ASCII format. The names derived from this root will use *trace-folder* as prefix and information about the node as suffix. Default value is *'owl-trace'*.

`-n-clients=n` integer value, specifies the number of OWL clients that will take part in the simulation. Default value is 1.

`-t-simulation=t` integer value, specifies the amount of time that will be simulated, in seconds. Default value is 10 s.

`-net-speed='net-speed'` string value, specifies the network speed of the CSMA bus. Possible values are *'100Mbps'* and *'1000Mbps'*. Default value is *'100Mbps'*.

`-seed=seed-value` integer value, specifies the random seed to be used in the simulation. Two runs of the simulation using the same random key will generate identical results. If the argument `-run` is used, the value given for `-seed` is ignored. Default value is 16 807.

`-run=n-runs` integer value, specifies a run number for the simulation. Different run numbers internally use different random seeds. Two simulation runs using the same run number will generate the same results. The argument `-run` provides a more user friendly and simplified way to manage random seeds in the simulation runs. Default value is 1.

`-osiat-model='model-name'` string value, specifies the type of OSIAT model to be used in the simulation. Supported models are *exponential* and *weibull*. More detail information about these models can be found in Section 4.4.4. Default value is *exponential*.

`-lambda=l-value` double value, together with `-osiat-model='exponential'` it specifies the rate value, λ, for the exponential-based OSIAT model. Together with `-osiat-model='weibull'` it specifies the scale parameter, λ, for the Weibull-based OSIAT model. Default value in both situations is 1.0.

`-kappa=k-value` double value, when `-osiat-model='weibull'` is used, it defines the shape parameter, κ for the Weibull-based OSIAT model. Default value in this situation is 1.0.

`-thres=thres-value` double value, specifies the t_{thres} parameter, in seconds, for all the OSIAT models. Default value is $0.501\,\mathrm{s}$.

`-c-apt=c-value` double value, specifies the value for c_{apt} parameter for all the OSIAT models, in the range $(0, 1)$. Default value is 0.8.

5.3 Simulation results

Both exponential and Weibull-based OSIAT models have been tested and compared against empirical data from Open Wonderland TestBed 2 (OWTB2) by setting up and using two different simulation scenarios. The simulator described in Section 5.2.2; based on the models *OsIatExp*, *OsIatWei*, *Osps* and the ns-3 framework; has been used to generate the network traffic for this evaluation. The choice of the parameters from testbed OWTB2 is due to its hardware and software homogeneity, which led to more consistent parameter results across clients. All the simulations have used the same OSPS model, based on the formula described in Section 4.5.3, independently of the OSIAT model they were using.

The evaluation of each OSIAT model involved 10 OWL clients and only 1 server interconnected by a $100\,\mathrm{Mbps}$ CSMA bus / Ethernet network. The simulation was executed 20 times, each one using a different random seed via de `-run` parameter.

Tab. 5.1.: BW and packet rate for testbed and simulated OWL traffic

Session	Packets/s	Bandwidth (B/s)
10-client	3.85	1152.94
Sim-exp	3.73	1067.88
Sim-wei	4.82	1379.20

5.3.1 Simulation based on exponential-based OSIAT model

The first simulation scenario was based on the exponential-based OSIAT model (Section 4.4.4) with parameter rate $\bar{\lambda} = 4.76$. This λ has been calculated as the mean of the λ values in Table D.6, with $\sigma_\lambda = 0.1$. The rest of OSIAT parameters used were $c_{apt} = 0.78$ and $t_{thres} = 0.5$.

The bandwidth and packet rate are used for simplicity as metrics to compare the experimental and simulated traffic [CLW03]. The total bandwidth generated by this simulation per each one of the 10 clients was 1067.88B/s per client and the packet rate equal to 3.73. These figures are slightly below the bandwidth and packet rate per client observed in the 10-client OWTB2 session, with 1152.94B/s and 3.85 packet/s respectively. These figures are shown in Table 5.1.

The Figure 5.8 helps to determine the similarity between the empirical cumulative distribution function (ECDF)s for the empirical and simulated data in this simulation. The plot has been generated calculating the ECDF for the aggregation of all the OSIAT values from the empirical and simulated traffic. The smooth blue ECDF corresponding to the probability distribution of the simulated traffic falls below the red and more jagged one corresponding to the experimental network traffic.

5.3.2 Simulation based on Weibull-based OSIAT model

The second simulation scenario was based on the Weibull-based OSIAT model (Section 4.4.4) with parameter scale $\hat{\lambda} = 0.142$, and shape $\hat{\kappa} = 0.51$, each of them calculated as the mean of the λ (σ_λ=0.01) and κ values ($\sigma_\kappa = 0.03$) in Table D.7. The rest of OSIAT parameters used were $c_{apt} = 0.78$ and $t_{thres} = 0.5$.

The total bandwidth generated by this simulation per each one of the 10 clients was 1379.20B/s per client and the packet rate equal to 4.82. These figures are slightly above the bandwidth and packet rate per client observed in the 10-client OWTB2 session, with 1152.94B/s and 3.85 packet/s respectively. These figures are shown in Table 5.1.

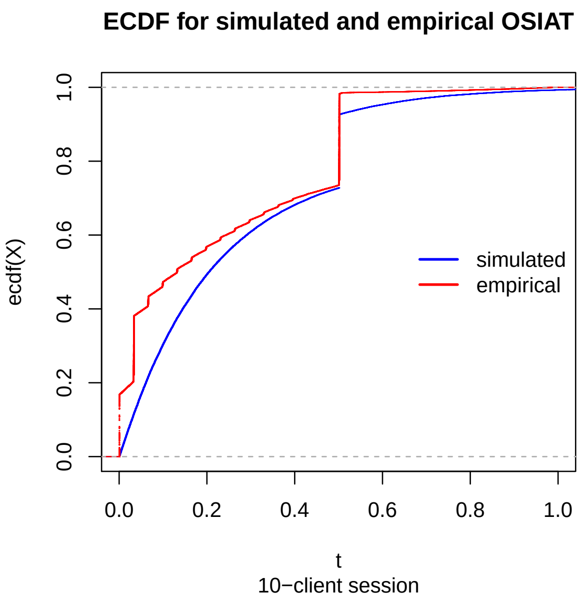


Fig. 5.8.: Exponential-based OSIAT model vs. empirical OSIAT

The Figure 5.9 helps to determine the similarity between the ECDFs for the empirical and simulated data in this simulation. The plot has been generated calculating the ECDF for the aggregation of all the OSIAT values from the empirical and simulated traffic. The smooth blue ECDF corresponding to the probability distribution of the simulation traffic mostly overlaps the red and more jagged one corresponding to the experimental network traffic, with regions where the model line is above the experimental one, hence the higher bandwidth and packet rate observed.

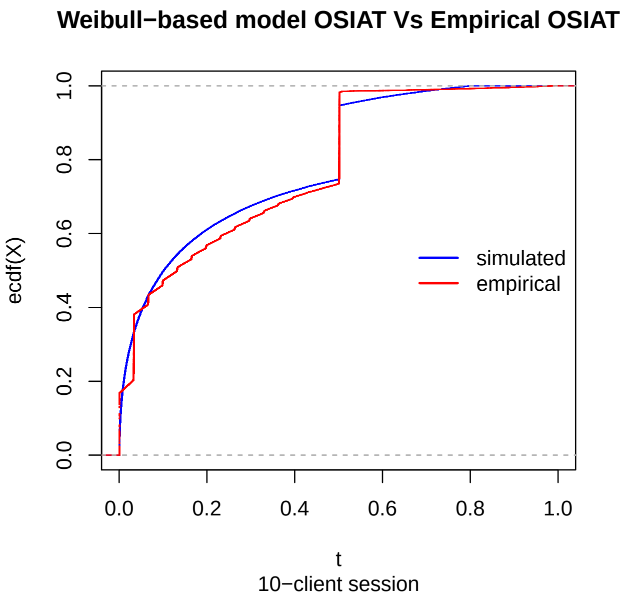


Fig. 5.9.: Weibull-based OSIAT model vs. empirical OSIAT

5.3.3 Conclusions

Observing the preliminary simulation results in Table 5.1 and the comparison of ECDFs in Figures 5.8 and 5.9, it is possible to conclude that both the exponential and Weibull-based OSIAT models together with the OSPS model provide a good approximation to the behaviour of OSIAT values on the experimental testbed.

5.4 Summary

The simulation tools presented in this chapter put into use and validate the OSIAT models for OWL generated using the analysis and modelling pipeline for NVE traffic proposed in previous chapters. The chosen tool for implementing the simulations has been ns-3, an open-source discrete-event network simulator written in C++ following OO paradigm and easily extendable.

The implementation of the simulation comprises the definition of several OO classes representing entities such as statistical models, applications that consume or generate network traffic or auxiliary classes to automate and simplify the configuration of the entities within the simulated scenario. All these software components allow to run simulations based on the exponential and Weibull-based OSIAT models for OWL described in Section 4.4.4 and the OSPS model described in Section 4.5.3.

The main simulation executable articulates and configures all the elements that take part in the simulation scenarios, providing a CLI interface for final users which allow them to provide all kind of arguments to configure and control the execution of the simulation, such as the type of OSIAT model and its associated parameters, number of iterations, pseudo random seeds, output format and so on.

Based on the simulation results, both the exponential and Weibull OSIAT models for OWL traffic, in conjunction with the OSPS model, manage to generate simulated traffic that resembles the observed in experimental sessions on the testbeds, in terms of both bandwidth and packet rate. These results can be used to argue that the analysis and modelling pipeline proposed for NVE systems such as OWL generate valid models to describe the inter-arrival time (IAT) and packet size (PS) variables that describe their network flows at microscale level.

Conclusions and future work 6

I may not have gone where I intended to go, but I think I have ended up where I needed to be.

Douglas Adams, *The Long Dark Tea-Time of the Soul*

The present chapter contain the conclusions derived from the study of NVE network traffic as well as the lines of work that can be addresses as future work.

6.1 Conclusions

The main conclusions derived from the work on NVE network traffic performed in the present work are the following.

- NVE systems have a relevant role in many fields in a global and highly interconnected world. The historical overview of NVE systems in Section 1.1 provided a sense of the importance the paradigm in fields such as entertainment, training or e-learning. Moreover, it has been observed that current trends are shifting from NVE frameworks such as *OpenSimulator* or OWL towards adapting mainstream video games such as *Minecraft* as canvas to develop NVE systems for research and scientific purposes.

- The NVE characterisation defined in Section 2.1 captures all the elements and characteristics that define a NVE and has proven to be key for the literature review allowing to define the *analysis pipeline* proposed in Chapter 3.

- OWL matched the NVE characterisation and proved to be technically viable study case on which to build experimental testbeds Open Wonderland TestBed 1 (OWTB1) and OWTB2 from Section 2.3 to get the necessary network traffic captures to validate the *analysis pipeline*.

- The study of the literature on network traffic of video games that qualify as NVE helped to define the *analysis pipeline* from Chapter 3, which follows a *microscale modelling* approach focused on IAT and PS to describe the network traffic, also providing methods metrics to assess statistical discrepancy, such as Q-Q plots or $\hat{\lambda}^2$ discrepancy metric.

- The *analysis pipeline* defined in Chapter 3 applied to the data from testbeds described in Section 2.3 generated the results discussed in Chapter 4. The pipeline helped to define two candidate models for OSIAT described in Section 4.4.4, both split distributions based respectively on the exponential and Weibull distributions. The OSPS values followed a simple discrete distribution described in Section 4.5.3.

- The models proposed in Chapter 4 for OSIAT and OSPS were implemented in simulations based on ns-3, an open-source discrete-event network simulation framework. Simulation results confirmed the similarity between experimental network traffic from the testbeds and the traffic generated by he simulators, in both terms of bandwidth and packet rate, being the Weibull-based OSIAT model slightly closer to the original OWL network traffic.

6.2 Future work

There are several lines of work that can be further developed as continuation of the present work. On the one hand, it is possible to further extend the *analysis pipeline* for modelling NVE network traffic proposed in Chapter 3. This pipeline only addresses network traffic which packets present randomly distributed IAT and PS values, leaving cases with strong autocorrelation out of the scope. Exploring the feasibility of applying time-series models to describe NVE traffic can be an valuable addition to the pipeline, and may help to identify traffic dynamics that are not fully explored in the current literature about NVE or video games network traffic.

A second potential line of work is further work in metrics for statistical discrepancy. The present work partly relies on qualitative methods to assess this discrepancy, such as Q-Q plots, techniques that require the human expertise to be of any use. The metric $\hat{\lambda}^2$ has proven to be a promising quantitative metric but with some limitations about the information that can reveal and some weakness regarding its performance on situations like heavy tailed distributions. Thus, refining the measurement of statistical discrepancy of microscale network models can yield useful results for the modelling of network traffic.

Finally, due to the nature of the ns-3 simulations proposed in Chapter 5 and some of their technical features, like the control of the pseudo-random seeds that control the execution of the simulation, it is worth to explore the possibility of adapting these processes to run on high-throughput computing (HTC) environments to exploit their potential parallelization. These concepts are further discussed in Appendix A.

A NVE simulations and HTC environments

The present appendix deals with the particularities of the ns-3 simulations for networked virtual environment (NVE) network traffic presented in Chapter 5 and specifically the problems associated to software distribution and provisioning time and how they can be minimising by running these simulations in a high-throughput computing (HTC) environment assisted by modern software distribution services.

The content in this appendix is based on the author's experience as system administrator in the *Distributed Data Processing Unit* at *SURFsara*, the Dutch national high-performance computing and e-Science support center in Amsterdam, Netherlands.

A.1 Running NVE simulations in HTC environments

The NVE traffic simulations implemented and executed in Chapter 5 using the ns-3 framework were mono-core processes, meaning that the simulation was running in a single core, the default behaviour for ns-3 simulations unless developers decide to include parallelisation techniques. Moreover, the inclusion of arguments that allow to specify pseudo-random seeds to control the execution of the simulation (Section 5.2.2) allows to exactly reproduce runs of the simulation by providing the same seed or, more interesting from the point of view of distributing and parallelising the simulation runs, by executing multiple times the simulation making sure each run is different from the rest by providing a unique seed to each one.

Single-core processes and the possibility to differentiate each run of the simulation make these simulations based on ns-3 a suitable candidate to be executed in HTC environments [LH13], computing environments were available resources work together to achieve the execution of long-running tasks taking advantage of parallelisation to achieve high degree of throughput in the results generated. It must be noted that while loosely related fields, HTC and high-performance computing (HPC) are different in many ways, while HPC tasks are characterized as needing large amounts of computing power for short periods of time, the HTC is more interested in how many *jobs* can be completed over a period of time than in how fast those jobs are individually.

Depending on the scale of available computing resources, the HTC environment can be a traditional cluster where all the machines are in the same geographical location, orchestrated by a job scheduler such as *HTCondor* [Faj+15] or *SLURM* ([YJG03]). In case where the amount of resources is far the reach of individual institutions, it is possible to scale out HTC infrastructure by using *grid computing*, which can be viewed as a federation of individual HTC cluster distributed around remote geographical locations, sharing resources to achieve common goals. This federation is accomplished by using a grid middleware software that orchestrates and distributes the workload all over the *grid* infrastructure. An example of grid middleware is *gLite*, used by the CERN LHC experiments and other scientific domains. It was implemented by collaborative efforts of more than 12 different academic and industrial research centers in Europe and it has been deployed in more than 250 institutions [And+08].

A.2 Software distribution in HTC environments

One of the technical challenges to solve when computing in distributed environments is the distribution of data and the software to process them. A classical approach has been provisioning the worker nodes (nodes that perform the actual computation in the HTC infrastructure) with local copies of the software required to process the data. While straightforward, this approach has the disadvantage of increasing the complexity of maintaining the worker nodes. More software means more maintenance work for the operators of the infrastructure, who have to be up to date with security alerts, apply security patches and bug fixes, make sure updates do not break the system or cause regressions. Moreover, this means each worker node has to be prepared beforehand to run jobs that require some specific software, reducing the flexibility of the infrastructure to host new workloads. In the specific case of simulations based on ns-3, system administrations have two options when provisioning worker nodes, on the one hand they can install the software from source code, making sure the binaries they generate are compatible with the target system and all the dynamic libraries required by these binaries are present on each node. On the other hand, some GNU/Linux distributions (the most extended operating system in the HPC/HPC environments) provided packages for ns-3 in their official repositories.

A more flexible approach for software distribution is making the job locally install the required software, using scripts that automate the download, configuration and compilation (if necessary) of the software. In the case of ns-3, in addition to the transfer time required to fetch the source code from the repositories of the ns-3 project or a local proxy (being the second option highly advised to avoid generating an accidental distributed denial-of-service (DDoS) attack by having hundreds of machines with high speed connections requesting copies of the repository), each job will have to spend a considerable amount of time compiling its instance of ns-3. The experimental measurements performed in this study show that compiling ns-3.29 in an machine equipped with an Intel i5-4690 CPU at 3.5GHz takes up to $20\,\mathrm{min}$ and up to 1GB RAM when using a single dedicated core at $100\,\%$ load. The reason to limit the compilation tests to one single core is because ns-3 simulations are single-core processes by default and they will likely be encapsulated into single-core HTC jobs to ensure an adequate resource assignment. In this approach, the ns-3 binaries generated cannot be directly reused between jobs, even if they run on the same worker node.

A previous approach can be improved by distributing pre-compiled ns-3 binaries compatible with the software base of the worker nodes. Thus, jobs can fetch these binaries from a local repository within the cluster, or they can be shared through a network file system. Traditional network file systems such as *NFS* are an alternative to implement this approach, but there are other solutions specifically designed and optimised for the distribution of software in HPC and HTC environments, such as CernVM File System (CVMFS), a read-only POSIX network file system based on the HTTP protocol that can benefit from using HTTP caching proxies such as *Squid* [Prof] for better performance and provides authentication and authorisation based on X.509 standard for public key certificates [Agu+08] [Wei+17].

Distributing pre-compiled software from a repository or a shared network file system still has to deal with the problem of library dependencies in the target worker node where the job is going to run. Different versions of the same software may depend in different versions of the same dynamic library, which may not be trivial in many Linux/Unix systems. One traditional solution to this problem has been using tools that manage the shell environment to make this possible, such as *Environment Modules* [FO96]. While a widespread solution, it is not specially robust and require a good understanding of the library dependencies of the software and the shell environment where software will run.

Recent years have witnessed the popularisation of container technologies as an efficient and platform-agnostic way to distribute and deploy software on heterogeneous systems. Containers encapsulate and isolate one or various processes from the rest of the host where they are running. This level of isolation is not as strong as in virtual machines, but enough to make the software within the container totally independent from the user space software of the host while being much more lightweight in resource terms than virtual machines. Thus, containers provide a true homogeneous runtime environment isolated from the particularities of the host, such as package manager, hardware devices or library versions, and a convenient way to distribute software in the form of images than can run on any GNU/Linux host with the required container runtime.

While *Docker* is becoming a *de-facto* standard for containers [Ber14], it does not fully match the security requirements to run on *grid computing* environments. This has motivated the creation of *Singularity*, a container technology developed with federated, research and grid infrastructures in mind [KSB17] [ADS17]. Some of the features that make Singularity ideal for HTC and grid computing are running as non-privileged user to minimise security risks, capability of directly using Docker images or converting them to Singularity Image Format (SIF), support for image registries and the possibility to distribute images as single files, making straightforward their execution on hosts supporting Singularity containers.

A.3 Summary

This chapter discussed the suitability of running the NVE simulations implemented in Chapter 5 on distributed HTC environments. These simulations can be encapsulated in HTC jobs and distributed across this kind of infrastructures.

It is also discussed the technical challenge posed by software distribution in distributed environments. Making the software available on the worker nodes is key to make possible running the simulations anywhere on a distributed infrastructure. In the case of ns-3, the compilation time is specially time consuming, so different alternatives to achieve the goal *"compile once, run many times"* are described.

The chapter also illustrates how the distribution of binaries using optimized network file systems such as CVMFS together with tools such as *Environment Modules* are giving way to more modern distribution channels like the container technologies. In this context, *Singularity* is becoming a standard in the academic and research context.

Testbed Appendix B

This appendix details the software tools and configuration used in Open Wonderland TestBed 1 (OWTB1) and Open Wonderland TestBed 2 (OWTB2), the experimental testbeds defined in Section 2.3.

B.1 Traffic capture with Wireshark and tshark

The packet capture, filtering and export to comma-separated value (CSV) has been performed using *Wireshark* [Tea19b], an open-source packet analyzer, and its command-line interface (CLI) counterpart *tshark* [Tea19a], more suitable for scripting and automation. Wireshark has been used to capture all the incoming and outgoing traffic. These raw packets has been stored in format `pcap` and later filtered and processed with both Wireshark and tshark. The object synchronisation inter-arrival time (OSIAT) traffic coming from a specific IP address can be filtered using Wireshark with a filter expression as follows:

```
tcp contains "MOVE" && ip.src==192.168.1.2
```

These kind of filters can be also used with *tshark*:

```
tshark -r raw_traces.pcap \
    -R "tcp contains "MOVE" && \
        ip.src==192.168.1.2 && \
        frame.time_relative >= 20 && \
        frame.time_relative <=220" \
        -w filtered_traces.pcap
```

The above *tshark* command filters all the OSIAT packets from the given *.pcap* file that have been originated in the client with IP *192.168.1.2*, selecting only those captured after 20 s and no later than 220 s.

The conversion from the `pcap` format, which contains most of the raw packet information, to plan-text CSV can be performed using Wireshark, by opening the pcap file, filtering the desired packets and proceeding to save the results with the menu option *Export packet dissections*, selecting CSV as output format.

B.2 AutoHotKey scripts

AutoHotKey (AHK) [Foua] is a free, open-source utility for Windows aimed to task automation. AHK can define and record macros which group keystrokes and mouse clicks in order to automate tasks in the form of scripts. These scripts can also be compiled into a Windows executable file which can be launched even if the host system does not have AHK installed.

AHK has been used in the clients of OWTB2 to generate patterns of keyboard keystrokes as input for the Open Wonderland (OWL) clients, thus replacing human players during the test gaming sessions. The script code for AHK (v1.1.09.00 - November 7, 2012) can be found below. It defines a *funcKey* function which receives up to two keystroke codes to be injected as real keystrokes and the amount of time that the keys must be pressed, simulating this way the *keydown* events to generate constant translation of the avatar. The script also includes several keybindings to help with the control of the automation process, triggering events and perform keystroke injection. It must be noted that the scripts counts with the presence of the OWL client launcher, and audio output configured to be redirected to the microphone input.

```
; Receives up to two keycodes and simulates the keydown event during a specifici
; amount of time
;       time: time the key is pressed
;       key1: first keycode
;       key2: second keycode (optional)

funcKey(time, key1, key2=0){
        ; activation of Shift modifier by default
        Send, {Shift Down}
        Send, {\%key1\% Down}

        ; checking the existence of a
        ; second valid parameter
        if key2<> 0
                Send, {\%key2\% Down}

        ; amount of time the keys have
        ; to be pressed
        Sleep time

        ; activating the release event
        ; for the pressed keys
        Send, {\%key1\% Up}

        if key2<> 0
                Send, {\%key2\% Up}
        Send, {Shift Up}
}

; Main body of the script, executed by default. The code defines several
; keybindings to automate several actions: launching OWL, broadcast audio, ...

; F1 launches the OWL client software. It requires the launcher being in a
; known path for the script
F1:: Run, "Wonderland.jnlp"

; F2 starts music play. The OS is in charge of redirecting the audio output
; to the mic input. For this script, VLC player has been used as audio source,
; using a preconfigured playlist
F2:: Run, "vlc.exe" —Z —L "music.xspf"

; F3 stars avatar movement
F3::

; OWL window gets focus
WinWait, Wonderland,
IfWinNotActive, Wonderland, , WinActivate, Wonderland,
WinWaitActive, Wonderland,

; loop to send keystrokes to the OWL window
Loop{

        ; each movement is performed a random amount of time Random
    ; function canbe seed using the command Random, , NewSeed
        Random, time, 0,3000
        Random, meanTime, 0,400

        ; selecting movement option to perform in addition to up, down, left
    ; and right, two diagonal movements have been included, they result from
    ; the combination of two keys pressed at the same time
        Random, action, 0, 5

        if GetKeyState("F4","p")
                break

        if action = 0
                funcKey(time, "Up")
        else if action = 1
                funcKey(time, "Left")
        else if action = 2
                funcKey(time, "Right")
        else if action = 3
                funcKey(time, "Down")
        else if action = 4
                funcKey(time, "Up", "Left")
        else if action = 5
                funcKey(time, "Up", "Right")
        Sleep, %meanTime%
}
return

; F5 stops the script
F5:: exitApp

return
```

B.3 HotKeyNet scripts

HotKeyNet (HKN) [Teaa] is a free multi-boxing software aimed for gamers wanting to control several in-game characters/avatars at the same time over different computers. HKN allows to control several programs through the network using a client-server architecture.

The HKN instance deployed on OWTB2 consisted in a HKN in server mode running in one of the OWL clients for Windows (*operator*) and instances of HKN in client mode in the rest of OWL clients. The *operator* machine itself also played the role of OWL client.

The script code for HKN (build 210) can be found below. It remotely controls and launches/stops the individual AHK scripts of the OWL client machines. Thus, a centralised *owl* computer can be used to trigger the AHK scripts through the network. The next script define an identification label for each OWL client IP address, broadcasting the F1-F5 keystrokes to all the OWL clients involved in the testing session.

```
// The operator machine also participates as
// OWL client machine, so the keystrokes
// must also be propagated locally

// Each label corresponds with the IP address
// of a OWL client from the testbed

<Label c241 local SendFocusWin >
<Label c199 10.1.15.199  SendFocusWin >
<Label c219 10.1.15.219  SendFocusWin >
<Label c238 10.1.15.238  SendFocusWin >
<Label c243 10.1.15.243  SendFocusWin >
<Label c245 10.1.15.245  SendFocusWin >
<Label c248 10.1.15.248  SendFocusWin >
<Label c249 10.1.15.249  SendFocusWin >
<Label c252 10.1.15.252  SendFocusWin >
<Label c253 10.1.15.253  SendFocusWin >


// F12 triggers the script

<Hotkey F12>
        // list of clients that receive
        // commands through the network
    <SendLabel c199, ... ,c253>
        // Each client has an instance
        // of the compiled form of the
        // AHK script, responsible of the
        // avatar movement
```

```
<Run "autohotkey_script.exe">


    // Capturing and propagating the F1—F5
    // keystrokes between the clients involved
    // in the gamins session
    <KeyList Comandos F1—F5>
    <Hotkey Comandos>
    // list of clients that receive
    // keystrokes through the network
    <SendLabel c241, ..., c243>
    <Key \%Trigger\%>
```

C

R implementation of OWL network traffic models and discrepancy metrics

This appendix contains the implementation in *R* language of the statistical models and discrepancy metrics for Open Wonderland (OWL) network traffic defined in Chapters 3 and 4.

C.1 Activity Correlation Parameter for OWL OSIAT models

R implementation of the C_{acp} parameter defined for the OWL object synchronisation inter-arrival time (OSIAT) models in Section 4.4.4.

```
#' Calculates Activity Correction Parameter (ACP).
#'
#' The activity rate of the player has an impact on the shape
#' of the IAT distibution. The ACP models this impact. Active
#' users present an ACP of 1, while inactive users have ACP ~0.9.
#' The ACP is calculated counting the proportion of packets
#' beyond the threshold (~0.5s) to determine the percentage of
#' values corresponding to the probability saturation around
#' 0.5 an tail values.
#' @param iat, vector with IAT values
#' @param intersection (i.e.  pexp/pweibul(0.49) value)
#' @return ACP value for the given IAT and associated session
#' @export
calculate.acp <- function(iat, intersection){
    e <- ecdf(iat)
    acp <- 1 - ((intersection - e(0.49))/intersection)
    return(acp)
}
```

C.2 Exponential-based OSIAT model for OWL

R implementation of the exponential-based OSIAT model for OWL defined in Section 4.4.4, Equation (4.3).

```
#' Cumulative Probability function for Truncated Exponential
#'
#' Calculates probability for given vector x using the
#' truncated exponencial distribution adjusted for
#' OSIAT modelling.
#' @param x vector of values which probability will be calculated
#' @param rate parameter for the exponential section
#' @param threshold delimiting the truncated distribution sections
#' @param acp, corrects acitivity-induced deviations
#' @return vector with the probability values
#' @export
pexp.trun <- function(x, rate, thres=0.5, acp=1){
    r <- sapply(x, pexp.trun.aux, rate=rate, thres=thres, acp=acp)
    return(r)
}


#' Auxiliary function for pexp.trun
#'
#' This function implements the pexp.trun logic, allowing to
#' use  sapply for better performance.
pexp.trun.aux <- function(x, rate, thres=0.5, acp=1){
    if(x < thres){
        r <- pexp(x, rate) * acp
    }else{
        r <- 1
    }
    return(r)
}


#' Quantile function for Truncated Exponential distribution
#'
#' @param q vector of quantiles to be calculated
#' @param exponential rate parameter for the exponential section
#' @param threshold delimiting the truncated distribution sections
#' @param acp, corrects acitivity-induced deviations
#' @return vector with quantile values
#' @export
qexp.trun <- function(q, rate, thres, acp){
    r <- sapply(q, qexp.trun.aux, rate=rate, thres=thres, acp=acp)
    return(r)
}


#' Auxiliary function for qexp.trun
#'
#' This function implements the qexp.trun logic, allowing to
#' use  sapply for better performance.
qexp.trun.aux <- function(q, rate, thres, acp){
    if(q < thres){
        y <- qexp(q/acp, rate)
    }else{
        y <- thres
    }
    return(y)
}


#' Random number generator for Truncated Exponential distribution
#'
#' Function uses inverse transforming sampling for the generation
#' of random numbers using the Truncated Exponential distribution
#' adapted for OSIAT.
#' @param n of random numbers to ge generated
#' @param exponential rate parameter for the exponential section
#' @param threshold delimiting the truncated distribution sections
#' @param acp, corrects acitivity-induced deviations
#' @return vector with random values
#' @export
rexp.trun <- function(x, rate, thres, acp){
    p <- runif(x)
    r <- qexp.trun(p, rate, thres, acp)
    return(r)
}
```

C.3 Weibull-based OSIAT model for OWL

R implementation of the Weibull-based OSIAT model for OWL defined in Section 4.4.4, Equation (4.4).

```
#' Cumulative Probability Function for Truncated Weibull
#'
#' Calculates probability for given vector x using the
#' truncated weibull distribution adjusted for
#' OSIAT modelling.
#' @param x vector of values which probability will be calculated
#' @param shape parameter for Weibull distribution section
#' @param scale parameter for Weibull distribution section
#' @param threshold delimiting the truncated distribution sections
#' @param acp, corrects acitivity-induced deviations
#' @return vector with the probability values
#' @export
pweibull.trun <- function(x, shape, scale, thres=0.5, acp=1){
    r <- sapply(x,
                pweibull.trun.aux,
                shape=shape,
                scale=scale,
                thres=thres,
                acp=acp)
    return(r)
}
```

```
#' Auxiliary function for pweibull.trun
#'
#' This function implements the pweibull.trun logic, allowing to
#' use  sapply for better performance.
pweibull.trun.aux <- function(x, shape, scale, thres=0.5, acp=1){
    if(x<thres){
        r <- pweibull(x, shape, scale) * acp
    }else{
        r <- 1
    }
    return(r)
}


#' Quantile function for Truncated Weibull distribution
#'
#' @param q vector of quantiles to be calculated
#' @param shape Weibull shape parameter for the Weibull section
#' @param scale Weibull scale parameter for the Weibull section
#' @param threshold delimiting the truncated distribution sections
#' @param acp, corrects acitivity-induced deviations
#' @return vector with quantile values
#' @export
qweibull.trun <- function(q, shape, scale, thres, acp){
    r <- sapply(q,
                qweibull.trun.aux,
                shape=shape,
                scale=scale,
                thres=thres,
                acp=acp)
    return(r)
}


#' Auxiliary function for qweibull.trun
#'
#' This function implements the qweibull.trun logic, allowing to
#' use  sapply for better performance.
qweibull.trun.aux <- function(q, shape, scale, thres, acp){
    if(q < thres){
        y <- qweibull(q/acp, shape, scale)
    }else{
        y <- thres
    }
    return(y)
}


#' Random number generator for Truncated Weibull distribution
#'
#' Function uses inverse transforming sampling for the generation
#' of random numbers using the Truncated Weibull distribution
#' adapted for OSIAT.
#' @param n of random numbers to ge generated
#' @param shape Weibull shape parameter for the Weibull section
#' @param scale Weibull scale parameter for the Weibull section
#' @param threshold delimiting the truncated distribution sections
```

```r
#' @param acp, corrects acitivity-induced deviations
#' @return vector with random values
#' @export
rweibull.trun <- function(x, shape, scale, thres, acp){
    p <- runif(x)
    r <- qweibull.trun(p, shape, scale, thres, acp)
    return(r)
}
```

C.4 Statistical discrepancy metrics

R implementation of the statistical discrepancy metrics λ^2 and $\hat{\lambda}^2$ described in Section 3.8.2. This implementation is distribution agnostic, receiving as parameter a couple of data sets, functions representing the cumulative distribution function (CDF) of a function or any combination of the former two. The functions will proceed to calculate the statistical discrepancy between the data sets or distributions.

```r
#' Calculates [hat] lambda square GoF metric for a given empirical
#' data set and Cumulative Distribution Function (CDF).
#'
#' The function measures the Goodness of Fit between the empirical
#' data and the proposed CDF. The parameters determine how the
#' lambda calculation is performed, as well as including the
#' parameters that determine the behaviour of the given CDF.
#' @param emp.data vector with empirical measurements
#' @param the.dist R function modelling the CDF for the
#'              theoretical distribution which GoF for the emp.data
#'              will be evaluated
#' @param breaks determines the number of "breaks" or bins to be
#'              used within the hat lambda square algorithm. -1
#'              indicates the function will calculate the optimal
#'              number of bins for the given parameters
#' @param mode switchs between standard lambda square and hat
#'              lambda square metrics
#' @param ... arbitrary list of parameters required by the
#'             the.dist function, i.e. exponential rate, Weibull's
#'             shape and shape
#' @return tuple of values ([hat] lambda square, var)
#' @examples
#'   > x <- rexp(1000, rate=5)
#'   > lambda.square(x, pexp, -1, "hat", rate=5)
#'   [1] 0.9155332 3.5994222
#' @export
lambda.square <- function(emp.data, the.dist, breaks, mode, ...){
        # calculate bin width [Pax94]
        min.aux <- min(emp.data)
        max.aux <- max(emp.data)

        if(breaks == -1){
                # optimal width according to [Bor00] and [Pax94]
                w <- 3.49 * sd(emp.data)*(length(emp.data))^(-1/3)
                # number of bins
                n.bins <- ceiling((max.aux - min.aux) / w)
                # fix the bin width
                w.new <- (max.aux - min.aux) / n.bins
                bins <- seq(min.aux, max.aux, by=w.new)
        } else {
                w <- (max.aux - min.aux)/breaks
                bins <- seq(min.aux, max.aux, by=w)
        }

        # calculate start and end for bin
        bin.a <- bins[-length(bins)]
        bin.b <- bins[-1]
        # calculated expected (theoretical) values
        z <- the.dist(bin.b, ...) - the.dist(bin.a, ...)
        # observed/empirical values
    y <- (hist(emp.data, breaks=bins, plot=FALSE))$counts;

        if(mode == "default") {
                if(min(z) == 0) {
                        y <- y[z!=0]
                        z <- z[z!=0]
                }
        } else {
                if(min(y) == 0){
                        y <- y[y!=0]
                        z <- z[y!=0]
                }
        }

        # number of observations and bins
        n.obs <- sum(y)
    n.bins <- length(z)
        # number of observations by theoretical frequencies
        # expected values according theoretical distribution
        npi <- n.obs * z;

        if(mode == "default") {
                x2 <- sum((y - npi)^2 / npi)
                K  <- sum((y - npi) / npi )
                D  <- y - npi
                E  <- npi
        } else {
        # this is hat lambda square
                x2 <- sum((npi - y)^2 /y)
                K  <- sum((npi - y) / y)
                D  <- npi -y
                E  <- y
        }

        n.param <- nargs() - 4
        df <- n.bins - 1 - n.param
        lambda2 <- (x2 -K -df) / (n.obs-1)
        # lambda-square variance
        T <- sum( ( D^3 -2*D*E + (5/2)*(D^2) + (3/2)*y) / (E^2) )
    aux <- (2*df + 4*n.obs*lambda2 + 4*n.obs*lambda2^2 + 4*T)
        var.lambda2 <-  aux / n.obs
        return( c ( lambda2, var.lambda2))
}
```

D

Derivatives from the analysis of Open Wonderland network traffic

D.1 ECDF and fitting CDFs for OSIAT

The present section contains the empirical cumulative distribution function (ECDF) plots for the object synchronisation inter-arrival time (OSIAT) values measured in the testing sessions from testbeds Open Wonderland TestBed 1 (OWTB1) and Open Wonderland TestBed 2 (OWTB2). Each ECDF plot is accompanied by the cumulative distribution function (CDF) curves for the proposed exponential and Weibull-based OSIAT models (Equations (4.3) and (4.4) respectively).

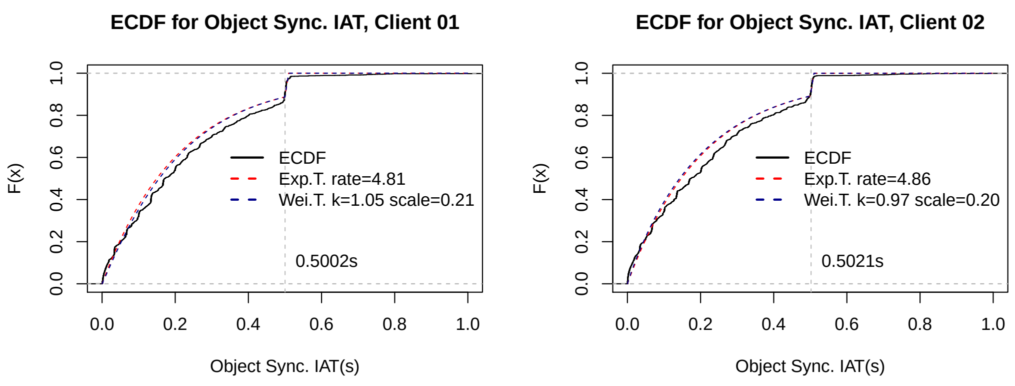


Fig. D.1.: ECDFs OSIAT values, 2-player session, OWTB1

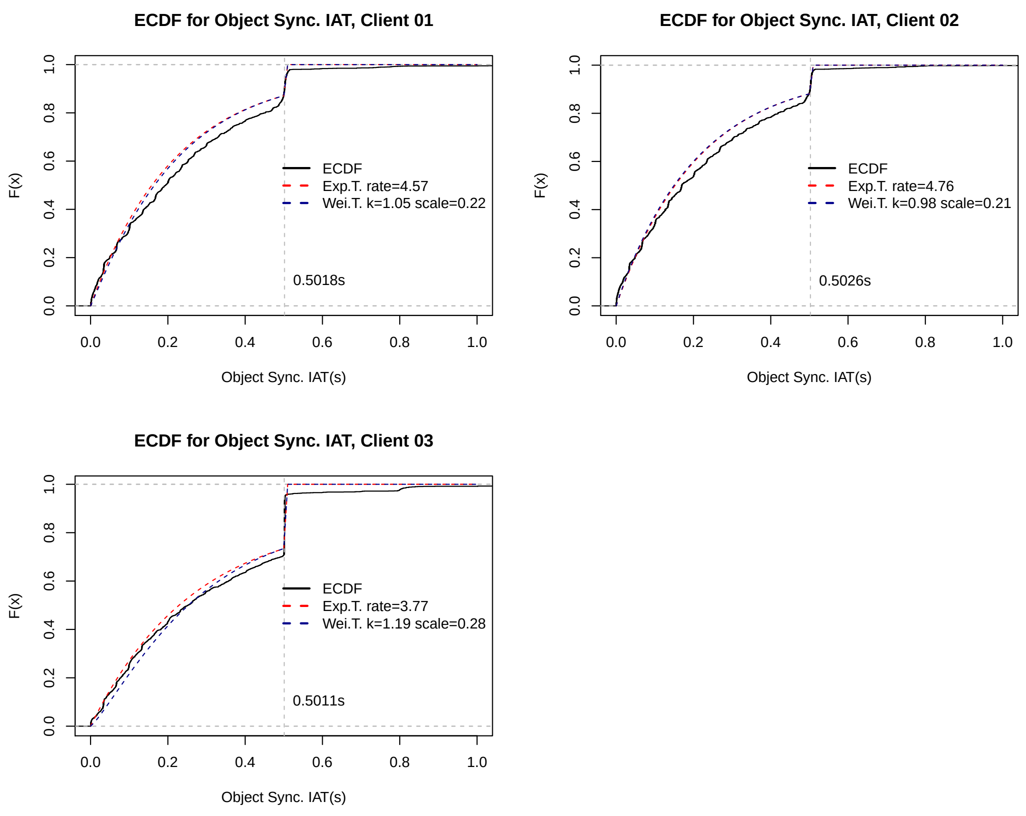


Fig. D.2.: ECDFs OSIAT values, 3-player session, OWTB1

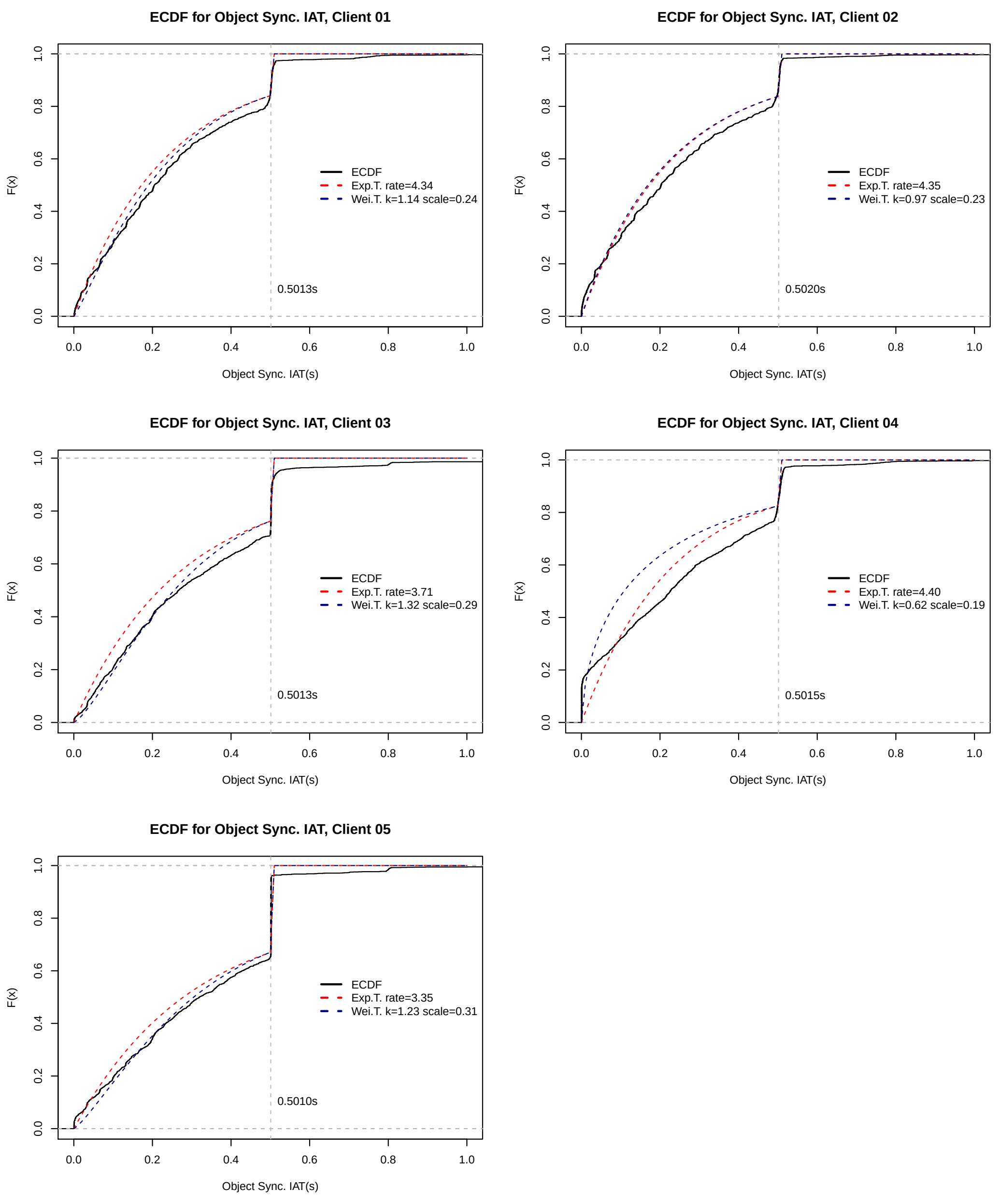


Fig. D.3.: ECDFs OSIAT values, 5-player session, OWTB1

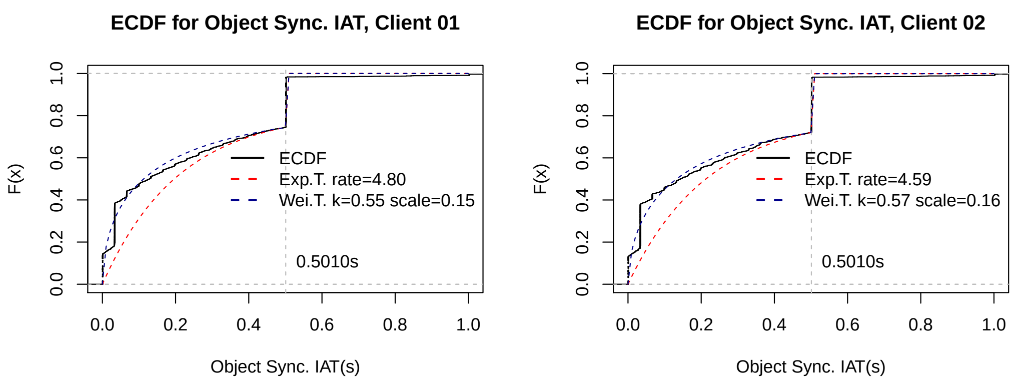


Fig. D.4.: ECDF plots for OSIAT, 2-player session, OWTB2

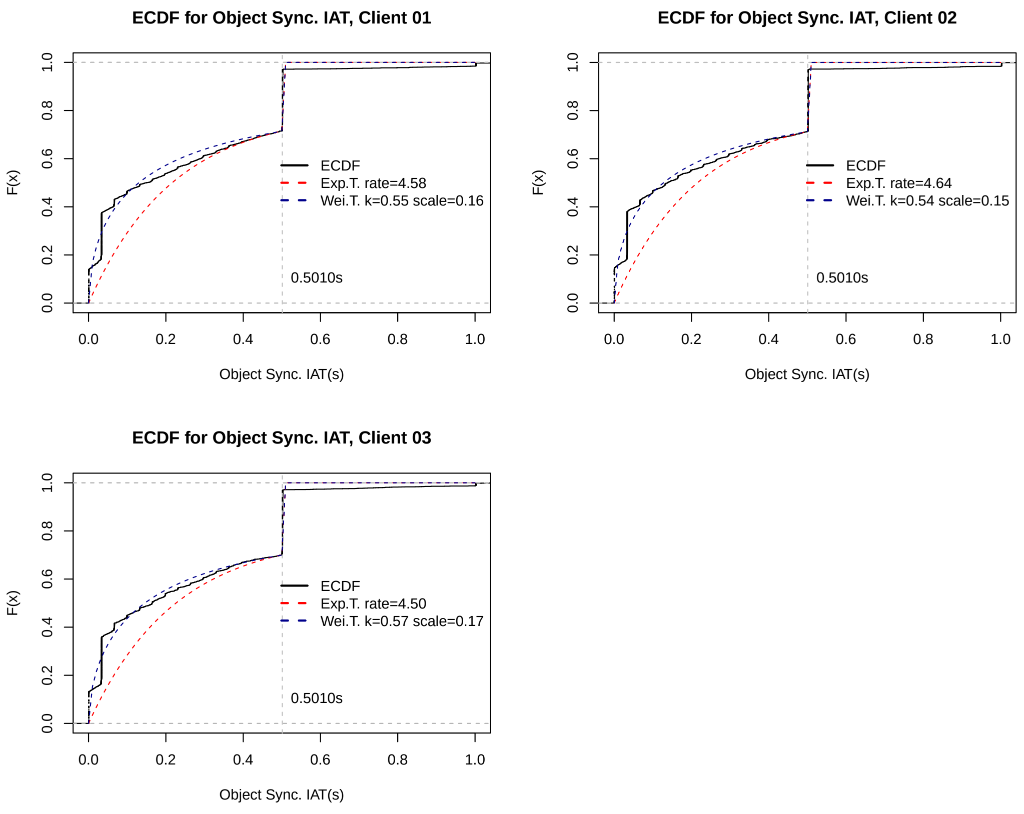


Fig. D.5.: ECDF plots for OSIAT, 3-player session, OWTB2

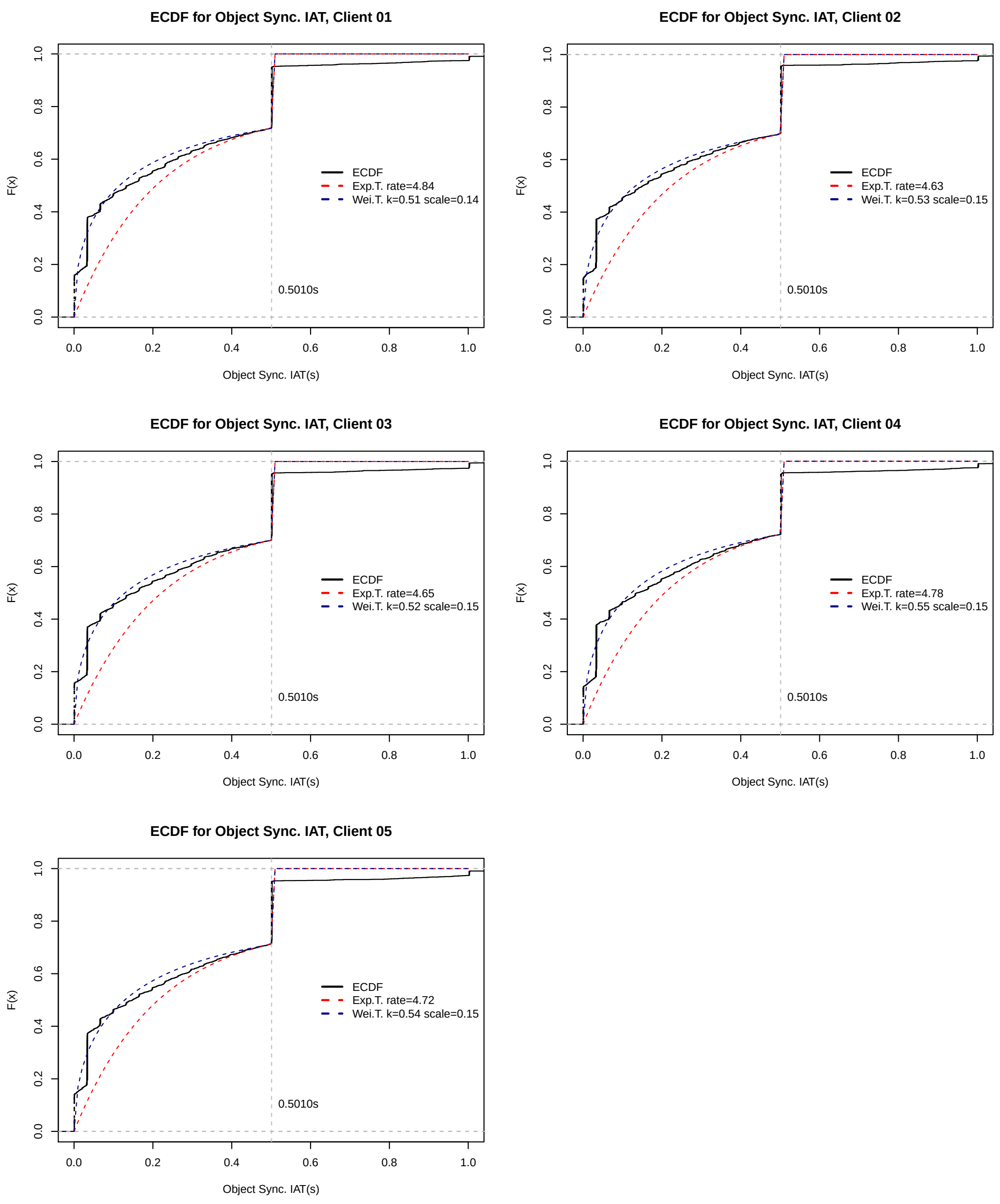


Fig. D.6.: ECDF plots for OSIAT, 5-player session, OWTB2

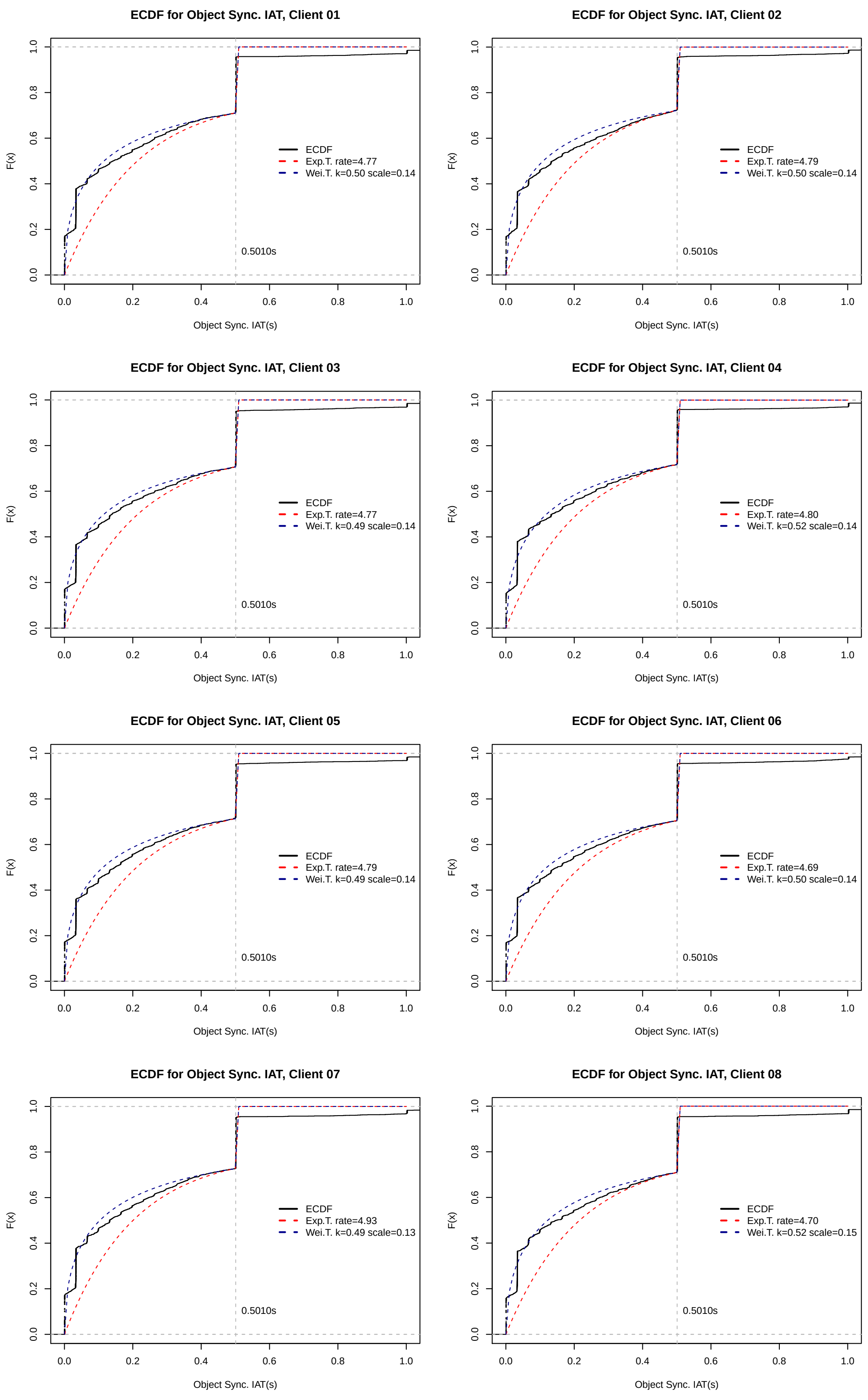


Fig. D.7.: ECDF plots for OSIAT, 8-player session, OWTB2

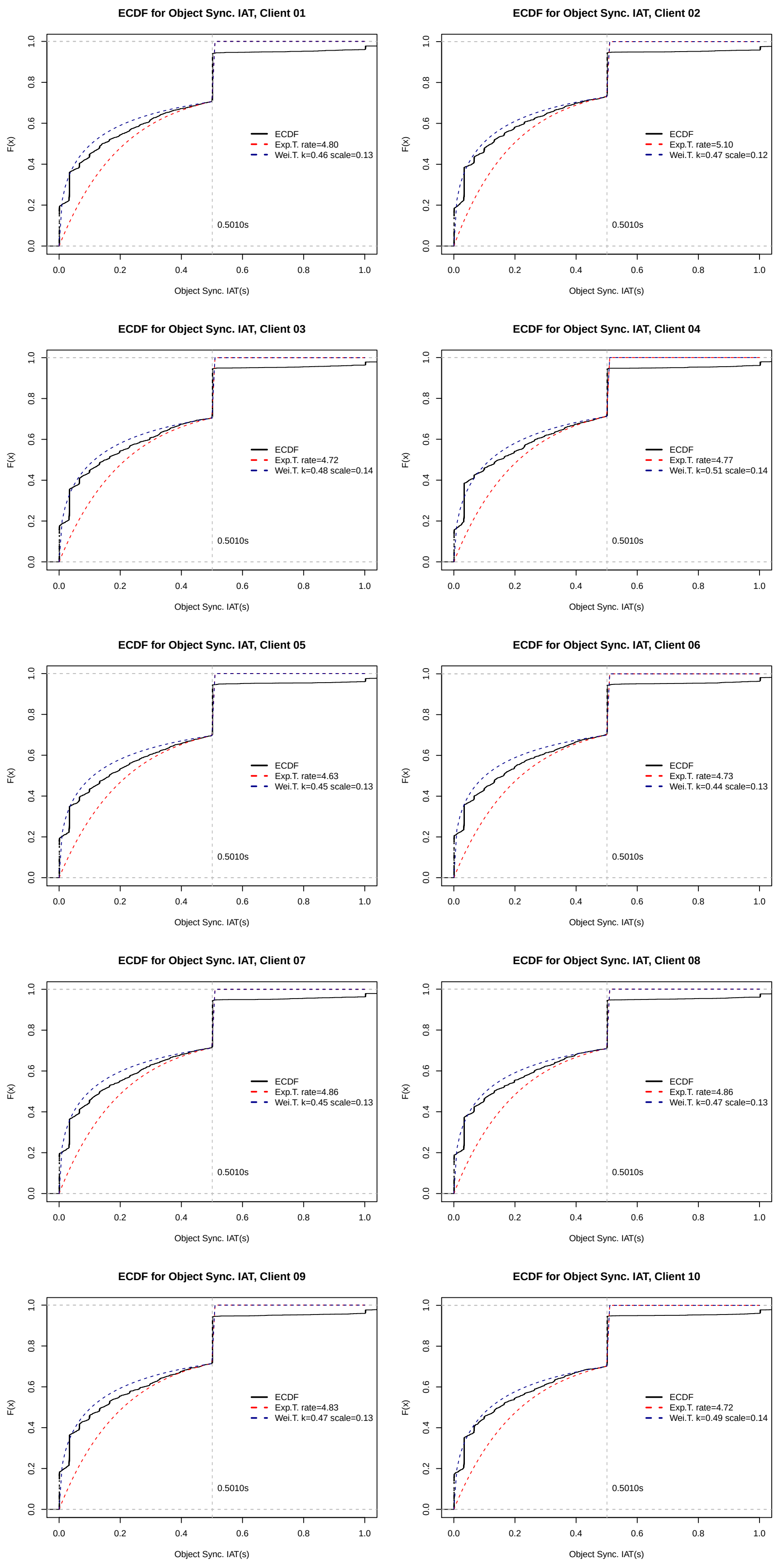


Fig. D.8.: ECDF plots for OSIAT, 10-player session, OWTB2

D.2 Autocorrelation functions for OSIAT

The present section contains the autocorrelation function (ACF) plots for the OSIAT values measured in the experimental sessions performed in the testbeds OWTB1 and OWTB2.

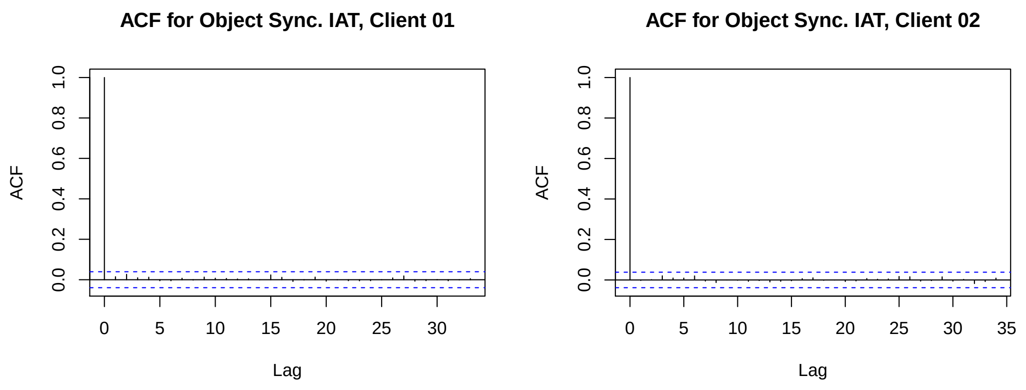


Fig. D.9.: ACF for OSIAT, 2-player session, OWTB1

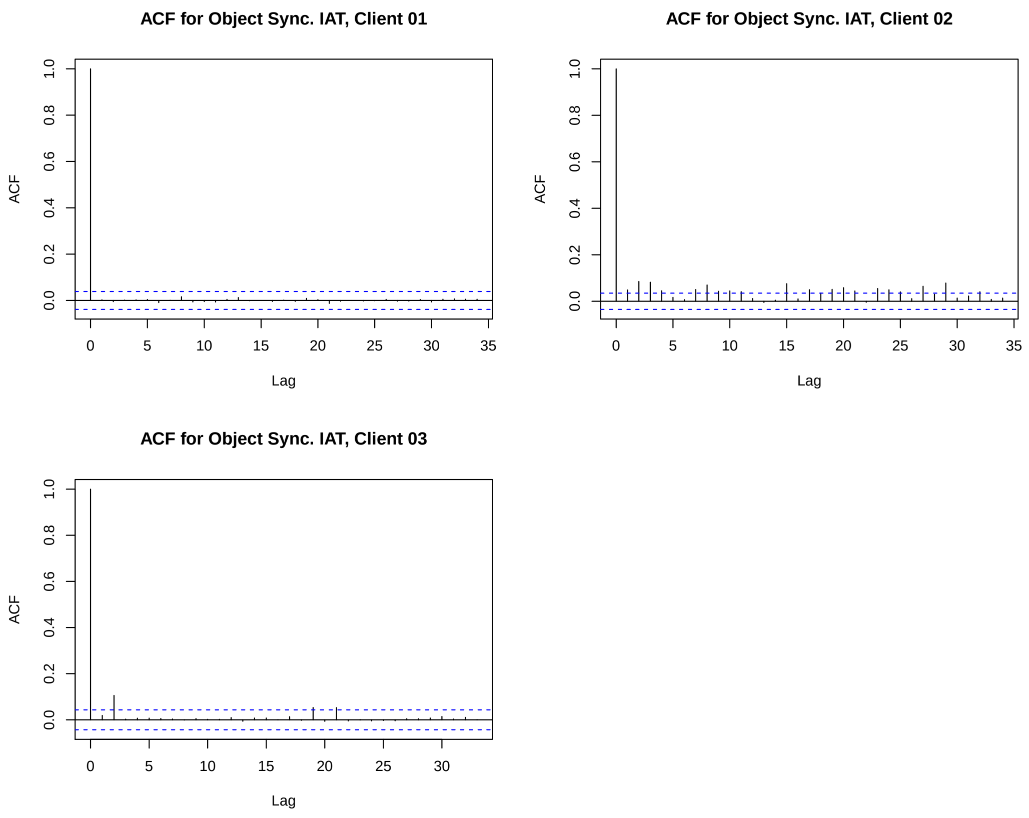


Fig. D.10.: ACF for OSIAT, 3-player session, OWTB1

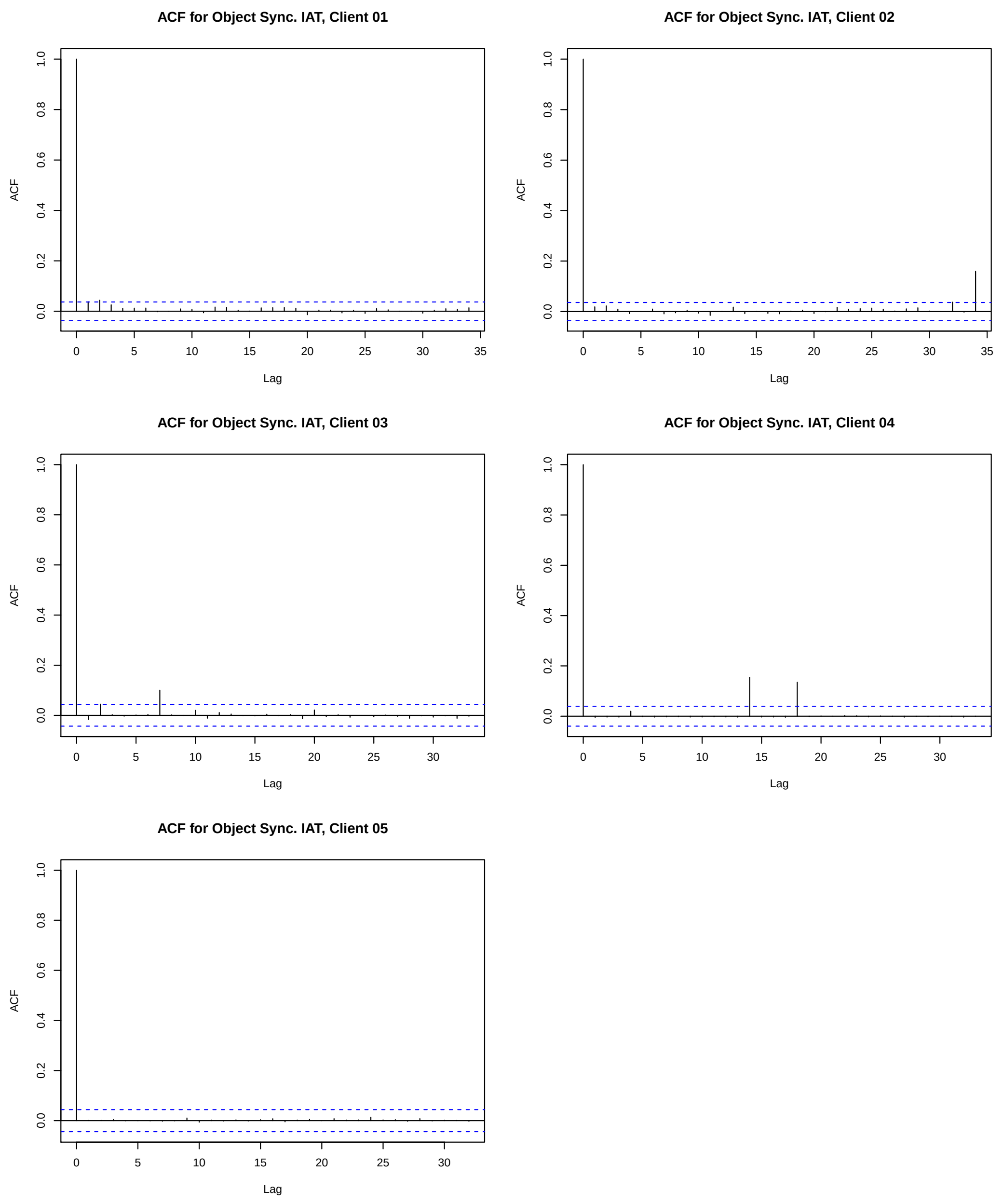


Fig. D.11.: ACF for OSIAT, 5-player session, OWTB1

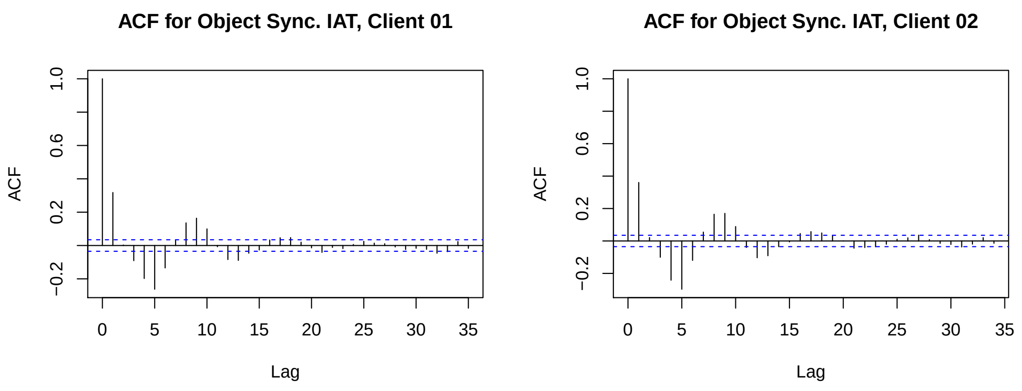


Fig. D.12.: ACF for OSIAT, 2-player session, OWTB2

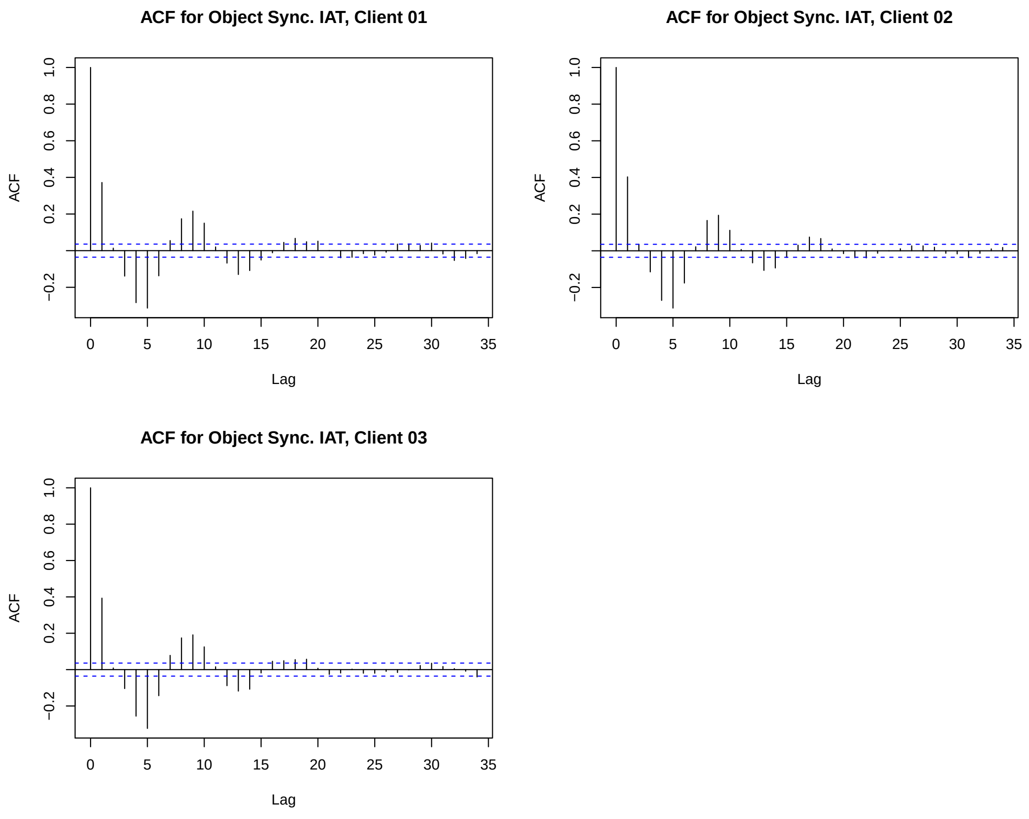


Fig. D.13.: ACF for OSIAT, 3-player session, OWTB2

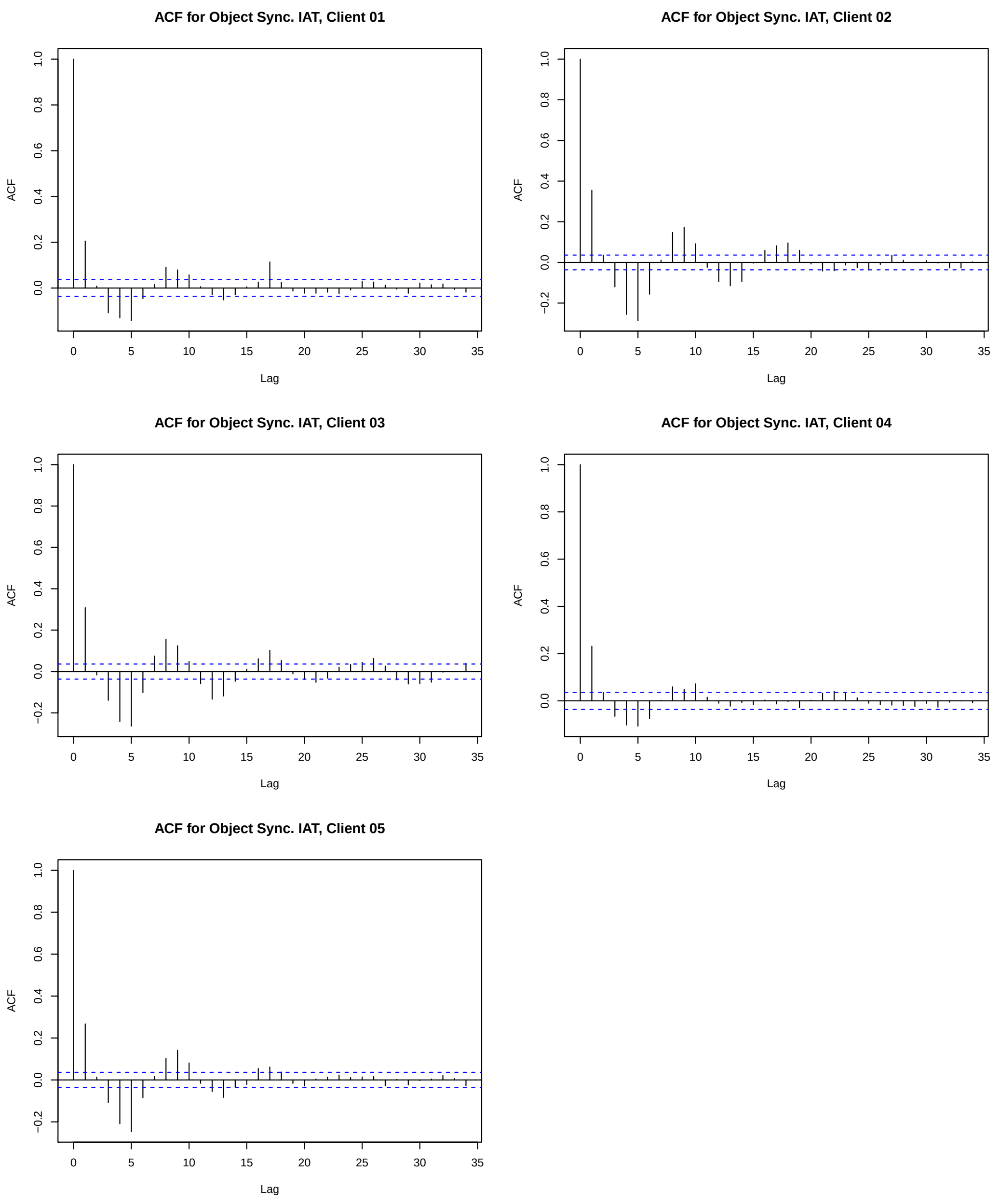


Fig. D.14.: ACF for OSIAT, 5-player session, OWTB2

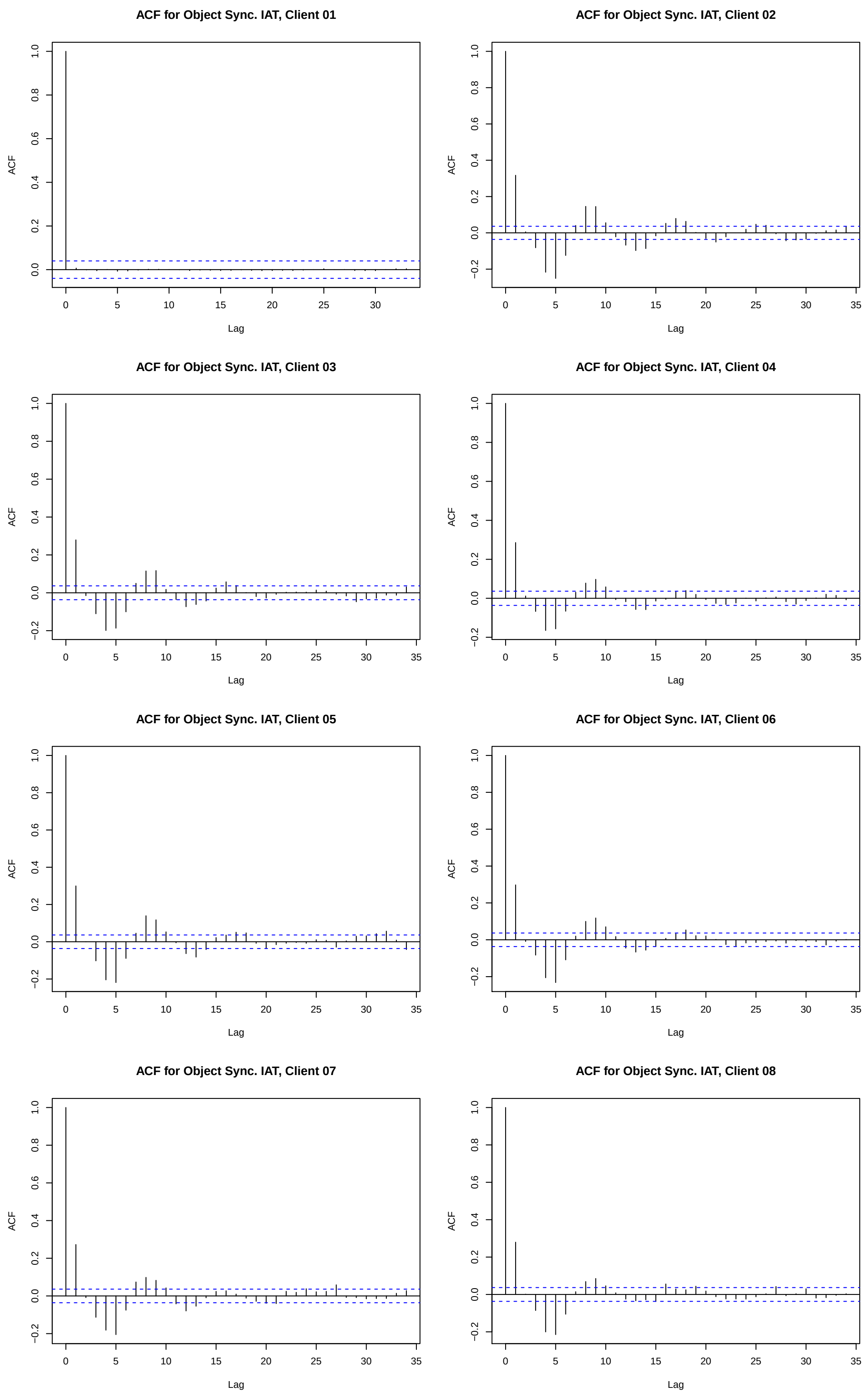


Fig. D.15.: ACF for OSIAT, 8-player session, OWTB2

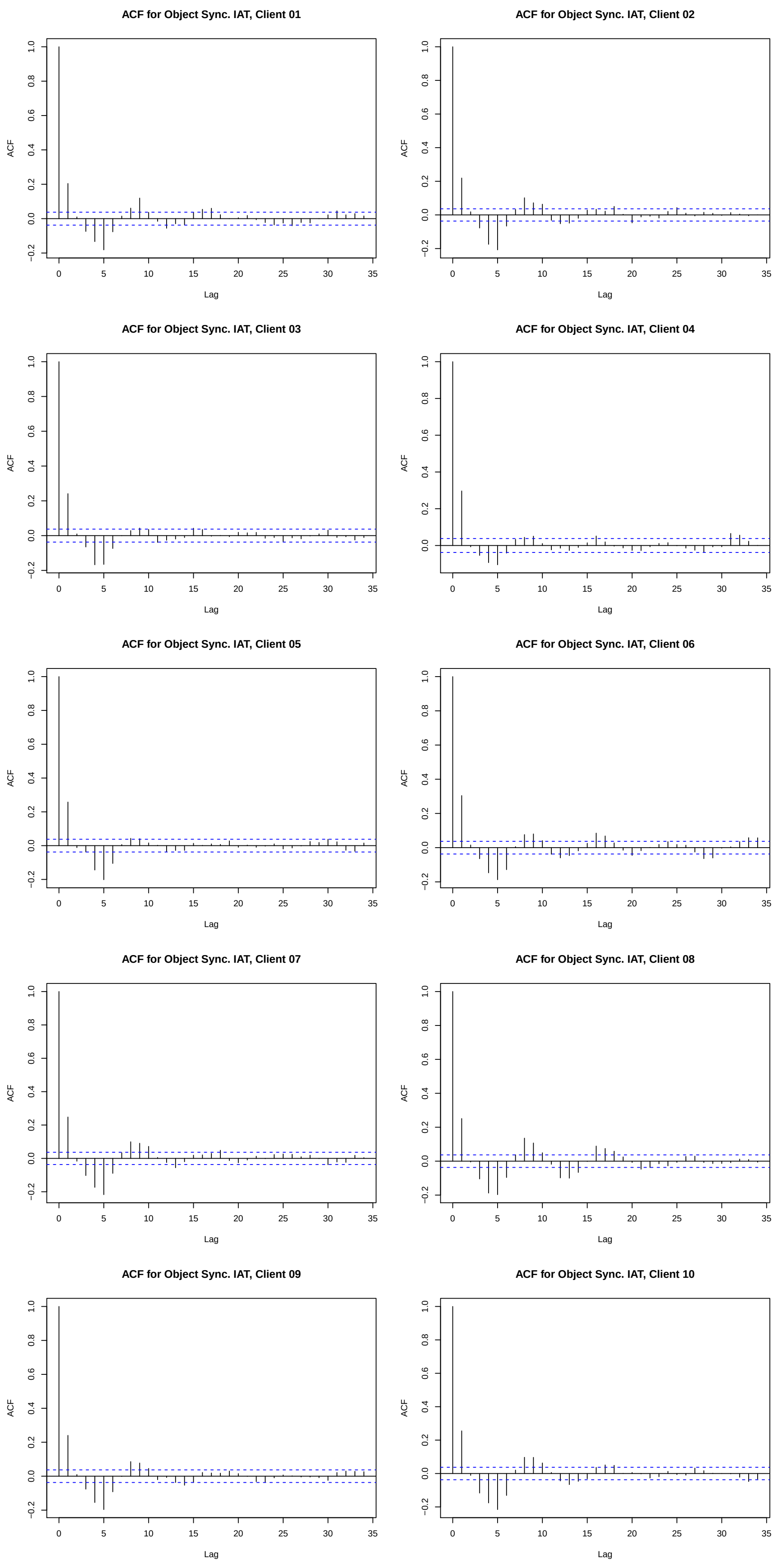


Fig. D.16.: ACF for OSIAT, 10-player session, OWTB2

D.3 Tail quantiles for OSIAT

The present section contains the Table D.1 showing the percentiles from Q_{95} to Q_{99} for the OSIAT values from OWTB2.

Tab. D.1.: Quantiles for OSIAT, OWTB2 sessions

Session	Client	$Q_{0.95}$	$Q_{0.96}$	$Q_{0.97}$	$Q_{0.98}$	$Q_{0.99}$
1-client	1	0.5011	0.5011	0.5011	0.5011	0.5012
2-client	1	0.5011	0.5011	0.5011	0.5014	0.8761
	2	0.5011	0.5011	0.5012	0.5015	0.8874
3-client	1	0.5011	0.5012	0.5014	0.8467	1.0020
	2	0.5011	0.5012	0.5021	0.8570	1.0021
	3	0.5012	0.5012	0.5039	0.7525	1.0020
4-client	1	0.5012	0.5012	0.6374	0.9036	1.0021
	2	0.5012	0.5014	0.7634	1.0012	1.0021
	3	0.5012	0.5012	0.6161	0.9030	1.0020
	4	0.5012	0.5016	0.7210	1.0015	1.0021
5-client	1	0.5016	0.6682	0.8812	1.0020	1.0022
	2	0.5012	0.6077	0.8323	1.0019	1.0021
	3	0.5013	0.6620	0.8820	1.0020	1.0021
	4	0.5013	0.6602	0.9151	1.0020	1.0021
	5	0.5013	0.7945	0.9563	1.0020	1.0026
6-client	1	0.5012	0.5914	0.8489	1.0020	1.0021
	2	0.5012	0.5828	0.8298	1.0018	1.0021
	3	0.5012	0.6014	0.9441	1.0020	1.0022
	4	0.5012	0.7122	0.9197	1.0019	1.0021
	5	0.5012	0.5991	0.9102	1.0019	1.0022
	6	0.5012	0.5044	0.8209	1.0015	1.0021
7-client	1	0.5013	0.6584	1.0014	1.0021	1.1838
	2	0.5013	0.7243	0.9919	1.0021	1.2692
	3	0.5029	0.7989	1.0020	1.0022	1.5029
	4	0.5013	0.7990	1.0020	1.0022	1.5030
	5	0.5013	0.6951	1.0019	1.0021	1.3579
	6	0.5012	0.6285	0.9846	1.0020	1.3066
	7	0.5017	0.7928	1.0010	1.0021	1.3921

Tab. D.1.: Quantiles for OSIAT, OWTB2 sessions

Session	Client	$Q_{0.95}$	$Q_{0.96}$	$Q_{0.97}$	$Q_{0.98}$	$Q_{0.99}$
8-client	1	0.5013	0.6898	0.9661	1.0021	1.4502
	2	0.5013	0.5842	0.9394	1.0020	1.2642
	3	0.5016	0.7414	1.0017	1.0021	1.2715
	4	0.5012	0.6075	0.9892	1.0020	1.2969
	5	0.5013	0.6614	1.0018	1.0021	1.4325
	6	0.5013	0.6814	0.9305	1.0020	1.4570
	7	0.5014	0.8151	1.0017	1.0022	1.4040
	8	0.5014	0.8001	1.0020	1.0022	1.3933
9-client	1	0.5013	0.7713	1.0009	1.0021	1.3308
	2	0.5055	0.8814	1.0020	1.0085	1.4663
	3	0.5013	0.7494	1.0020	1.0021	1.3609
	4	0.5013	0.7735	1.0019	1.0023	1.3463
	5	0.5074	0.8006	1.0019	1.0157	1.4069
	6	0.5013	0.6734	1.0007	1.0021	1.3046
	7	0.5013	0.6439	1.0019	1.0022	1.2888
	8	0.5088	0.7630	1.0011	1.0020	1.1466
	9	0.5013	0.6391	0.9940	1.0022	1.2887
10-client	1	0.7465	0.9848	1.0021	1.1544	1.5031
	2	0.6953	1.0018	1.0021	1.2069	1.5029
	3	0.5665	0.9171	1.0020	1.0625	1.5030
	4	0.6937	0.9626	1.0021	1.0417	1.4494
	5	0.5458	0.9747	1.0020	1.2116	1.5031
	6	0.5480	0.9319	1.0020	1.0022	1.5030
	7	0.6248	0.9014	1.0020	1.0584	1.4882
	8	0.6353	0.9501	1.0021	1.1182	1.5030
	9	0.6784	1.0009	1.0021	1.1058	1.5028
	10	0.6236	0.9826	1.0020	1.0817	1.5030

D.4 Activity Correction Parameter for OSIAT models

The present section contains the Tables D.2 and D.3 of *Activity Correction Factor* values (C_{acp}, Section 4.4.4) calculated for each one of the experimental sessions from OWTB1 and OWTB2 respectively.

Tab. D.2.: C_{acp} values for exponential and Weibull-based models, OWTB1

Session	Client	acp_{exp}	acp_{wei}
2-client	1	0.95	0.94
	2	0.96	0.97
3-client	1	0.94	0.93
	2	0.95	0.95
	3	0.83	0.81
5-client	1	0.91	0.90
	2	0.92	0.92
	3	0.84	0.81
	4	0.87	0.91
	5	0.79	0.78

Tab. D.3.: C_{acp} values for exponential and Weibull-base models, OWTB2

Session	Client	acp_{exp}	acp_{wei}
1-client	1	0.82	0.87
2-client	1	0.82	0.87
	2	0.80	0.85
3-client	1	0.79	0.84
	2	0.79	0.84
	3	0.78	0.83
4-client	1	0.79	0.84
	2	0.79	0.84
	3	0.79	0.84
	4	0.79	0.84
5-client	1	0.79	0.84
	2	0.77	0.82
	3	0.78	0.83
	4	0.79	0.84
	5	0.79	0.83

Tab. D.3.: C_{acp} values for exponential and Weibull-base models, OWTB2

Session	Client	acp_{exp}	acp_{wei}
6-client	1	0.77	0.82
	2	0.79	0.84
	3	0.78	0.84
	4	0.78	0.83
	5	0.79	0.85
	6	0.79	0.84
7-client	1	0.78	0.84
	2	0.79	0.84
	3	0.77	0.82
	4	0.78	0.83
	5	0.78	0.83
	6	0.79	0.85
	7	0.78	0.83
8-client	1	0.78	0.84
	2	0.79	0.85
	3	0.78	0.83
	4	0.79	0.84
	5	0.78	0.84
	6	0.78	0.83
	7	0.80	0.85
	8	0.78	0.84
9-client	1	0.79	0.85
	2	0.78	0.84
	3	0.79	0.84
	4	0.79	0.84
	5	0.78	0.84
	6	0.80	0.86
	7	0.78	0.83
	8	0.79	0.85
	9	0.80	0.85

Tab. D.3.: C_{acp} values for exponential and Weibull-base models, OWTB2

Session	Client	acp_{exp}	acp_{wei}
10-client	1	0.78	0.83
	2	0.79	0.85
	3	0.78	0.83
	4	0.78	0.84
	5	0.77	0.83
	6	0.77	0.83
	7	0.78	0.84
	8	0.78	0.84
	9	0.78	0.84
	10	0.77	0.83

D.5 MLE parameters for OSIAT models

The present section contains the Tables D.4 and D.6, which contain the values for λ rate for the exponential-based distributions calculated for OSIAT using maximum likelihood estimation (MLE) on the traffic captured in OWTB1 and OWTB2 respectively. Tables D.5 and D.7 contain the values for λ scale and κ shape for the Weibull-based distributions calculated using MLE on the traffic captured in OWTB1 and OWTB2 respectively.

Tab. D.4.: MLE exponential, λ rate for OWTB1 sessions

Session	Client	λ	σ	loglik
2-client	1	4.36	0.09	1165.42
	2	4.52	0.09	1336.87
3-client	1	3.83	0.08	881.08
	2	4.50	0.08	1582.61
	3	2.99	0.07	198.52
5-client	1	3.74	0.07	898.33
	2	3.93	0.07	1092.78
	3	2.68	0.06	-28.49
	4	3.15	0.06	360.67
	5	2.57	0.06	-111.05

Tab. D.5.: MLE Weibull, κ shape and λ scale for OWTB1 sessions

Session	Client	κ	σ_κ	λ	σ_λ	loglik
2-client	1	0.94	0.01	0.22	0.00	1174.80
	2	0.89	0.01	0.21	0.00	1367.35
3-client	1	0.86	0.01	0.24	0.01	950.09
	2	0.96	0.01	0.22	0.00	1587.37
	3	0.89	0.01	0.31	0.01	236.05
5-client	1	0.95	0.01	0.26	0.01	907.20
	2	0.88	0.01	0.24	0.01	1135.38
	3	0.87	0.01	0.34	0.01	27.47
	4	0.59	0.01	0.21	0.01	1194.90
	5	0.85	0.01	0.35	0.01	-30.30

Tab. D.6.: MLE exponential λ rate for OWTB2 sessions

Session	C	λ	σ	loglik
1-client	1	3.90	0.07	1015.6
2-client	1	4.49	0.08	1624.5
	2	4.33	0.08	1456.2
3-client	1	4.19	0.08	1305.9
	2	4.26	0.08	1379.5
	3	4.15	0.08	1265.6
4-client	1	4.20	0.08	1315.3
	2	4.13	0.08	1240.8
	3	4.26	0.08	1379.5
	4	4.21	0.08	1324.7
5-client	1	4.06	0.08	1174.4
	2	4.05	0.08	1164.2
	3	4.04	0.07	1148.5
	4	4.03	0.07	1145.0
	5	4.01	0.07	1116.9
6-client	1	4.03	0.07	1140.3
	2	4.18	0.08	1288.0
	3	4.13	0.08	1245.4
	4	4.06	0.08	1174.0
	5	4.14	0.08	1252.1
	6	4.17	0.08	1287.7

Tab. D.6.: MLE exponential λ rate for OWTB2 sessions

Session	C	λ	σ	loglik
7-client	1	4.05	0.08	1158.6
	2	4.09	0.08	1196.5
	3	3.83	0.07	947.4
	4	3.84	0.07	949.3
	5	3.91	0.07	1022.3
	6	4.08	0.08	1186.4
	7	3.99	0.07	1103.5
8-client	1	3.35	0.07	501.34
	2	4.08	0.08	1190.84
	3	3.99	0.07	1101.46
	4	4.02	0.07	1131.93
	5	4.02	0.07	1134.74
	6	3.97	0.07	1084.86
	7	4.10	0.08	1211.33
	8	3.93	0.07	1038.27
9-client	1	4.09	0.08	1202.88
	2	3.94	0.07	1053.50
	3	4.04	0.07	1148.73
	4	3.99	0.07	1097.91
	5	3.95	0.07	1063.74
	6	4.13	0.08	1246.69
	7	3.96	0.07	1067.68
	8	4.04	0.07	1153.53
	9	4.12	0.08	1232.84
10-client	1	3.78	0.07	895.47
	2	4.02	0.07	1134.87
	3	3.85	0.07	959.81
	4	3.73	0.07	852.60
	5	3.76	0.07	876.66
	6	3.86	0.07	977.42
	7	3.92	0.07	1037.01
	8	3.90	0.07	1016.04
	9	3.87	0.07	983.40
	10	3.83	0.07	940.43

Tab. D.7.: MLE Weibull, κ shape and λ scale for OWTB2 sessions

Session	Client	κ	σ_κ	λ	σ_λ	loglik
1-client	1	0.55	0.01	0.16	0.01	2230.3
2-client	1	0.54	0.01	0.16	0.01	2635.9
	2	0.57	0.01	0.17	0.01	2282.2
3-client	1	0.55	0.01	0.17	0.01	2219.7
	2	0.54	0.01	0.17	0.01	2358.7
	3	0.57	0.01	0.18	0.01	2042.2
4-client	1	0.53	0.01	0.16	0.01	2377.6
	2	0.55	0.01	0.17	0.01	2131.0
	3	0.55	0.01	0.17	0.01	2308.2
	4	0.54	0.01	0.17	0.01	2304.7
5-client	1	0.51	0.01	0.16	0.01	2330.0
	2	0.53	0.01	0.17	0.01	2172.5
	3	0.52	0.01	0.17	0.01	2198.2
	4	0.54	0.01	0.17	0.01	2111.8
	5	0.54	0.01	0.18	0.01	2046.2
6-client	1	0.53	0.01	0.17	0.01	2090.9
	2	0.52	0.01	0.16	0.01	2398.9
	3	0.50	0.01	0.16	0.01	2437.4
	4	0.54	0.01	0.17	0.01	2149.0
	5	0.51	0.01	0.16	0.01	2408.7
	6	0.53	0.01	0.17	0.01	2293.5
7-client	1	0.50	0.01	0.16	0.01	2333.1
	2	0.50	0.01	0.16	0.01	2434.3
	3	0.50	0.01	0.17	0.01	2045.8
	4	0.51	0.01	0.17	0.01	2064.8
	5	0.50	0.01	0.17	0.01	2171.2
	6	0.48	0.01	0.16	0.01	2543.8
	7	0.50	0.01	0.16	0.01	2275.6
8-client	1	0.48	0.01	0.16	0.01	1953.6
	2	0.49	0.01	0.16	0.01	2446.1
	3	0.49	0.01	0.16	0.01	2362.2
	4	0.52	0.01	0.17	0.01	2239.8
	5	0.49	0.01	0.16	0.01	2393.7
	6	0.49	0.01	0.16	0.01	2301.6
	7	0.49	0.01	0.16	0.01	2535.7
	8	0.51	0.01	0.17	0.01	2137.5

Tab. D.7.: MLE Weibull, κ shape and λ scale for OWTB2 sessions

Session	Client	κ	σ_κ	λ	σ_λ	loglik
9-client	1	0.48	0.01	0.15	0.01	2583.5
	2	0.49	0.01	0.16	0.01	2339.9
	3	0.48	0.01	0.16	0.01	2484.7
	4	0.50	0.01	0.16	0.01	2323.6
	5	0.44	0.01	0.15	0.01	2729.1
	6	0.44	0.01	0.14	0.01	2965.8
	7	0.49	0.01	0.17	0.01	2296.5
	8	0.48	0.01	0.16	0.01	2517.1
	9	0.49	0.01	0.15	0.01	2584.1
10-client	1	0.46	0.01	0.16	0.01	2391.4
	2	0.47	0.01	0.15	0.01	2670.6
	3	0.48	0.01	0.17	0.01	2238.8
	4	0.50	0.01	0.17	0.01	2004.4
	5	0.46	0.01	0.16	0.01	2339.2
	6	0.44	0.01	0.15	0.01	2633.3
	7	0.45	0.01	0.15	0.01	2615.5
	8	0.46	0.01	0.16	0.01	2492.2
	9	0.47	0.01	0.16	0.01	2375.1
	10	0.49	0.01	0.17	0.01	2181.8

D.6 Q-Q plots for OSIAT values

The present section contains a selection of Q-Q plots comparing the quantiles for the OSIAT values from OWTB1 and OWTB2 sessions against the fitting OSIAT models proposed in the present work.

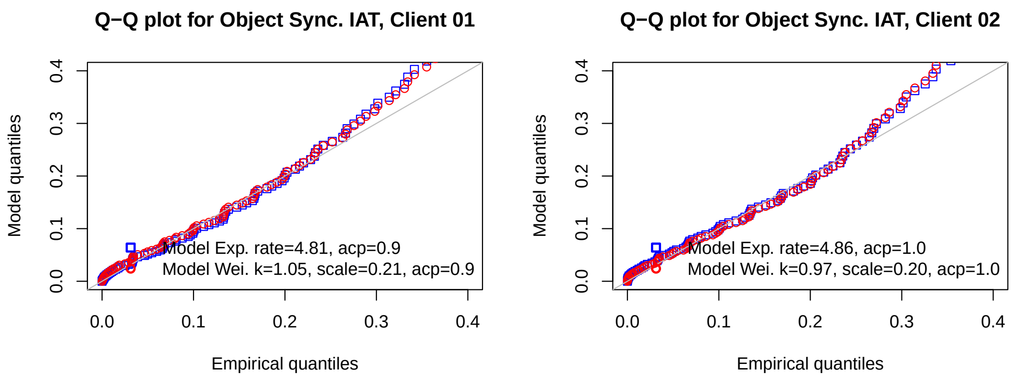


Fig. D.17.: Q-Q for OSIAT models, 2-player session, OWTB1

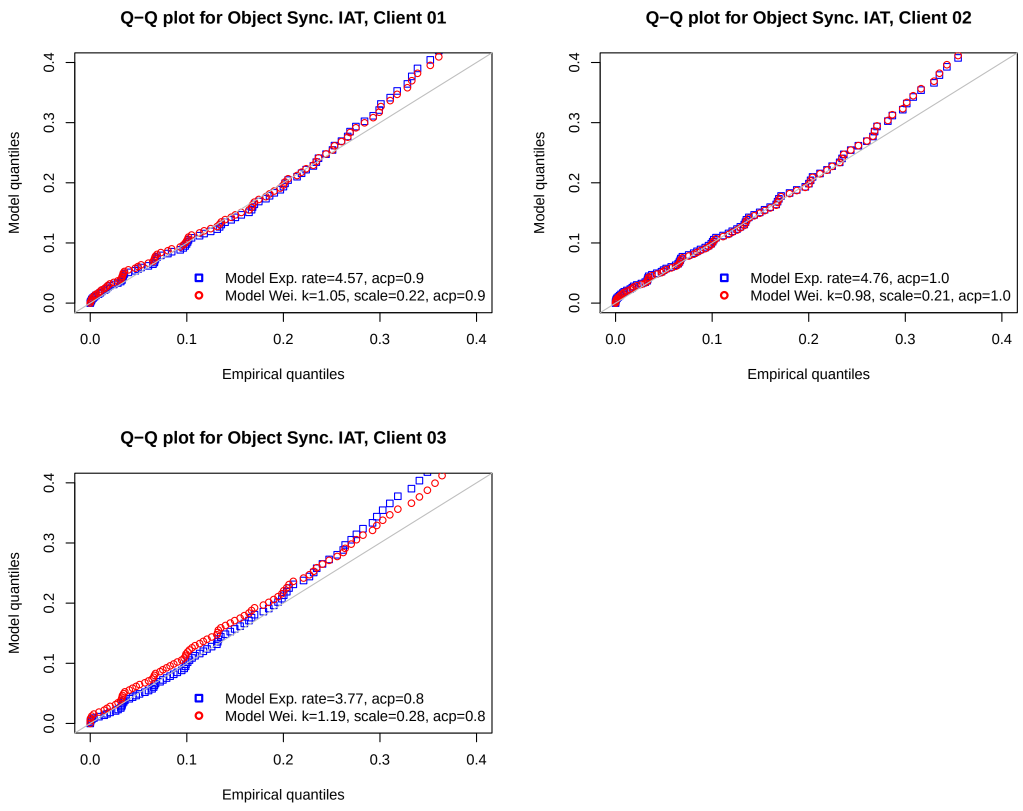


Fig. D.18.: Q-Q for OSIAT models, 3-player session, OWTB1

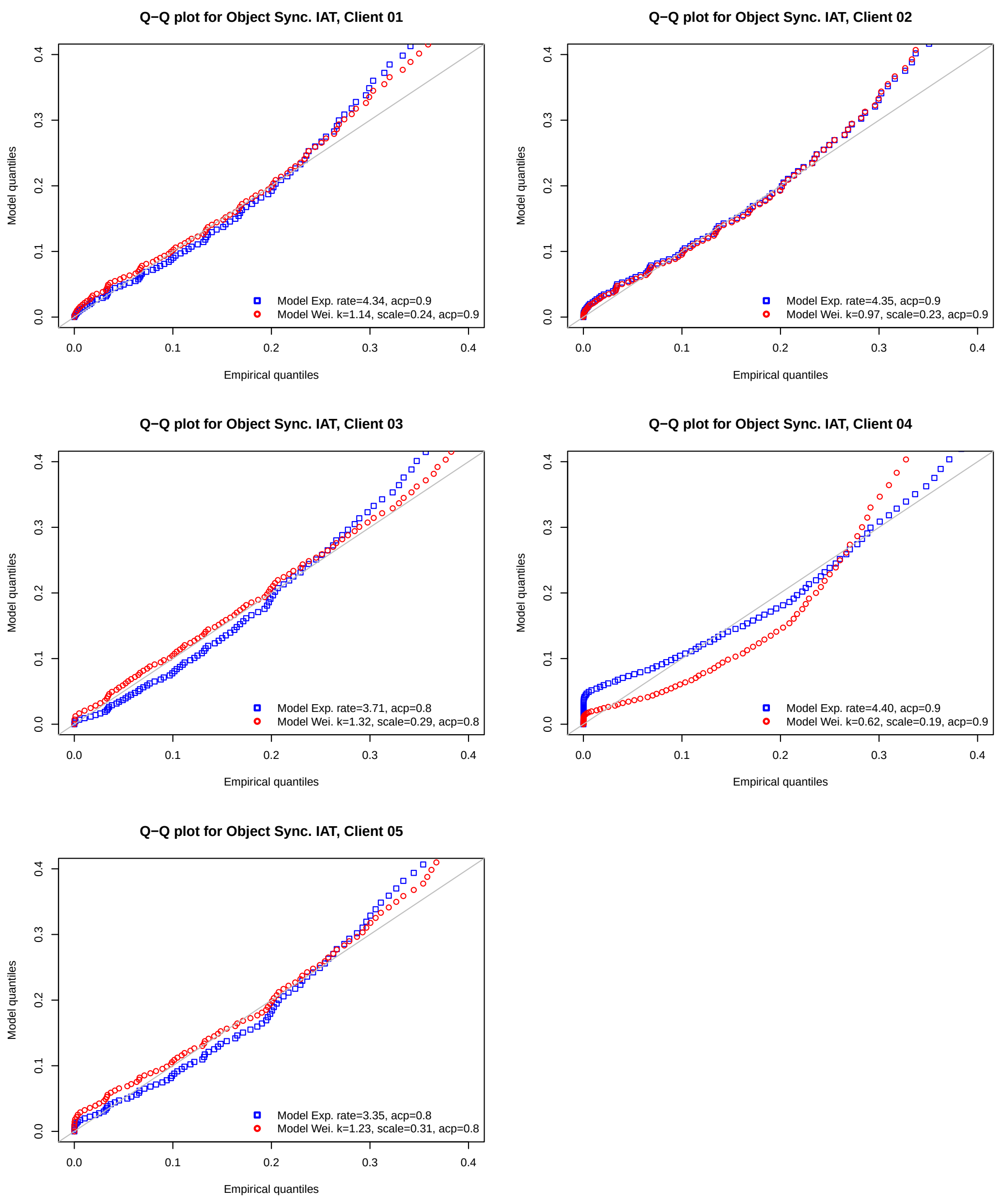


Fig. D.19.: Q-Q for OSIAT models, 5-player session, OWTB1

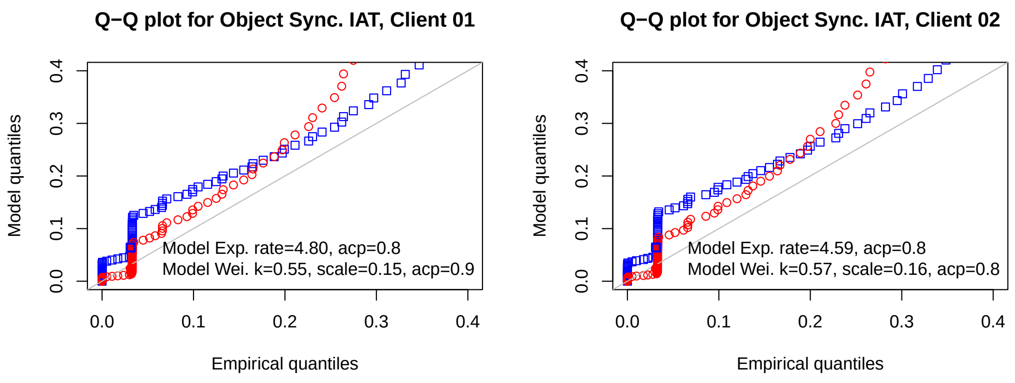


Fig. D.20.: Q-Q for OSIAT models, 2-player session, OWTB2

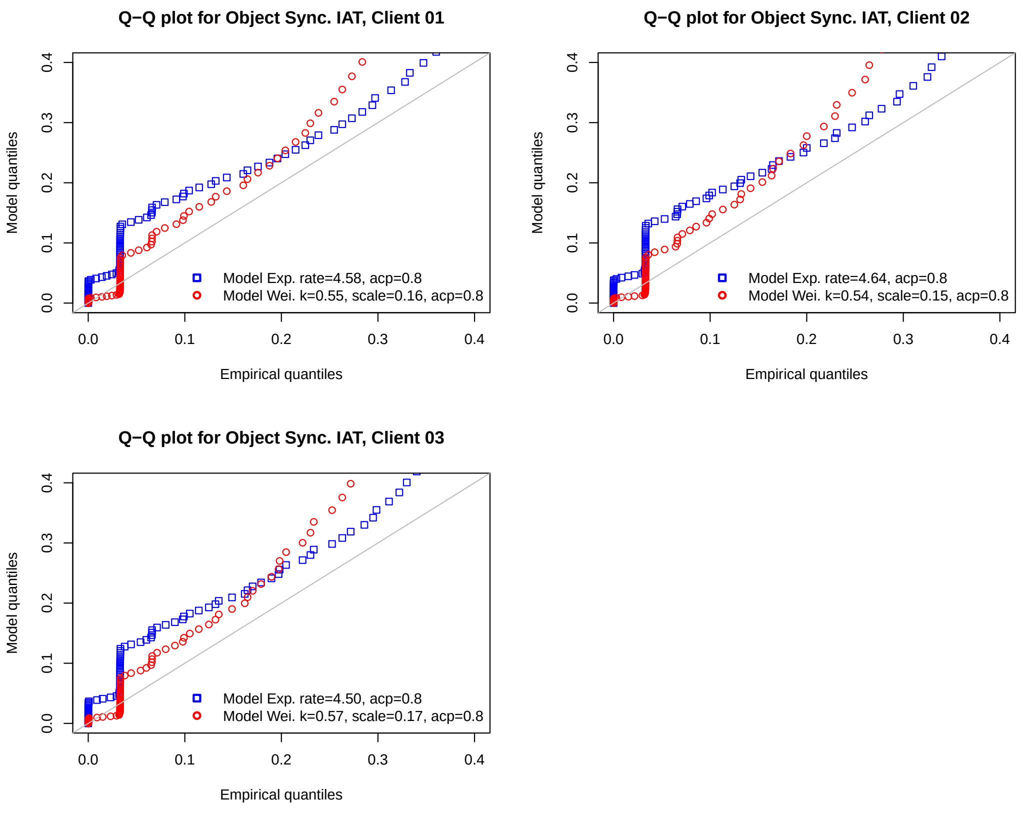


Fig. D.21.: Q-Q for OSIAT models, 3-player session, OWTB2

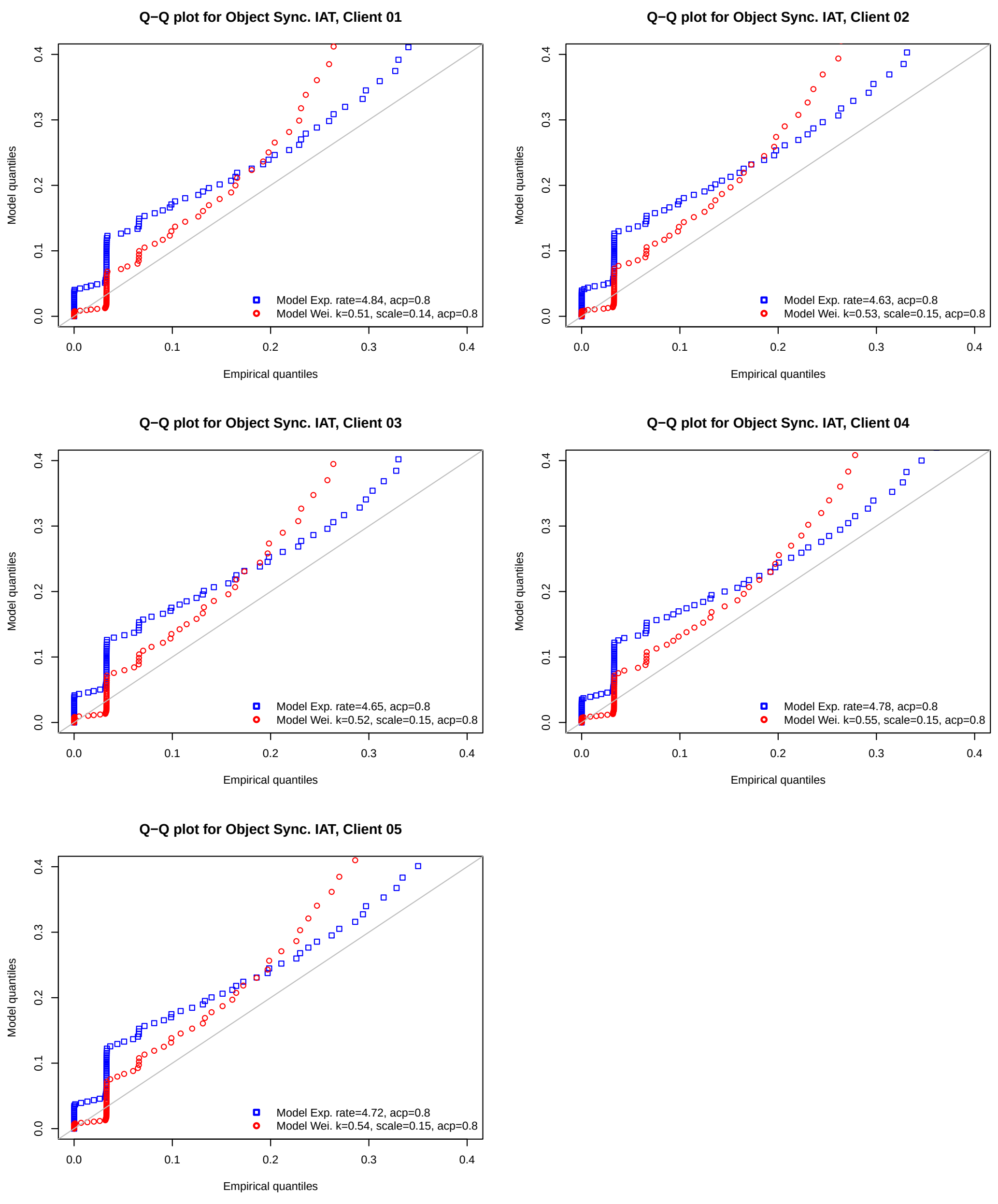


Fig. D.22.: Q-Q for OSIAT models, 5-player session, OWTB2

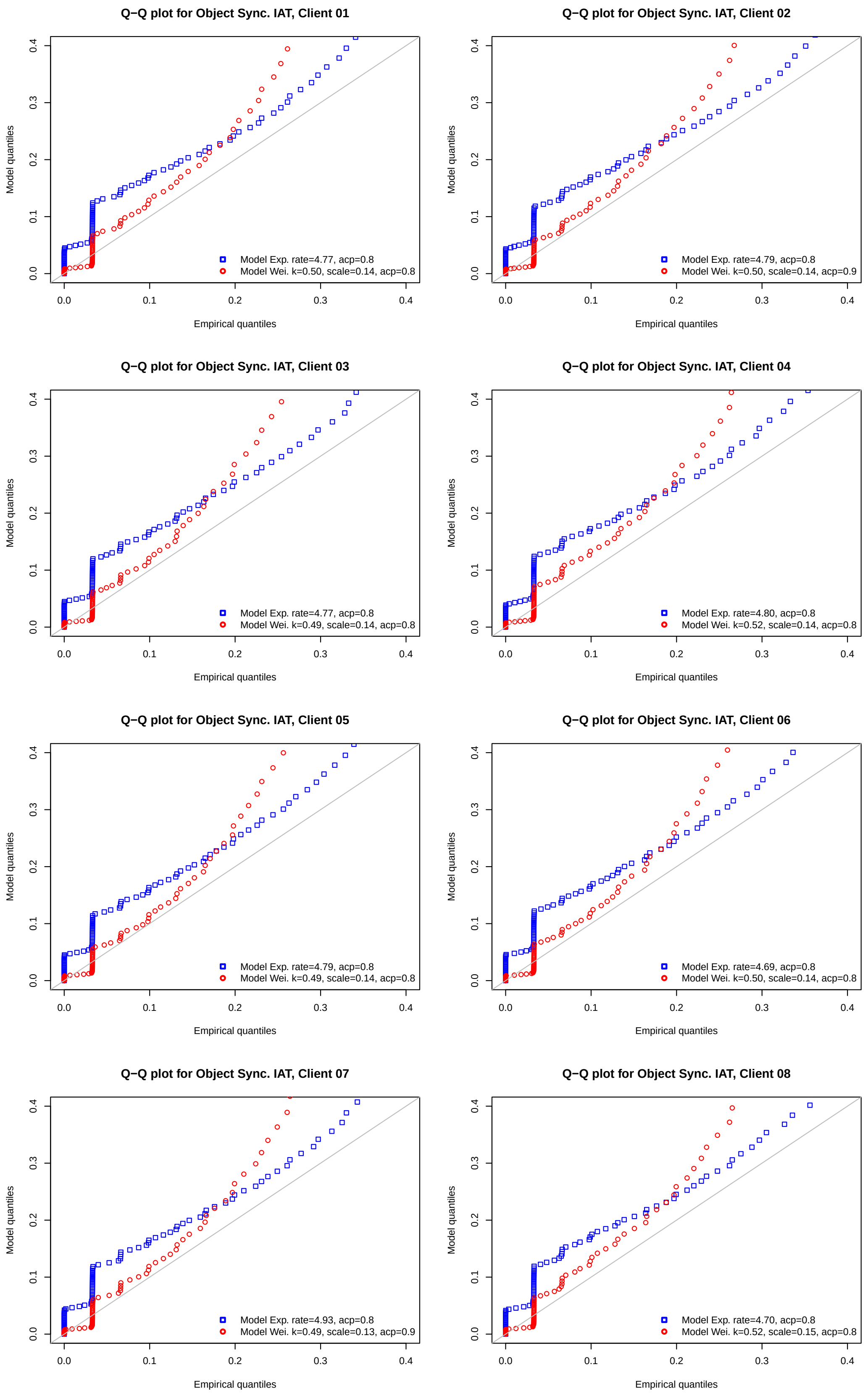


Fig. D.23.: Q-Q for OSIAT models, 8-player session, OWTB2

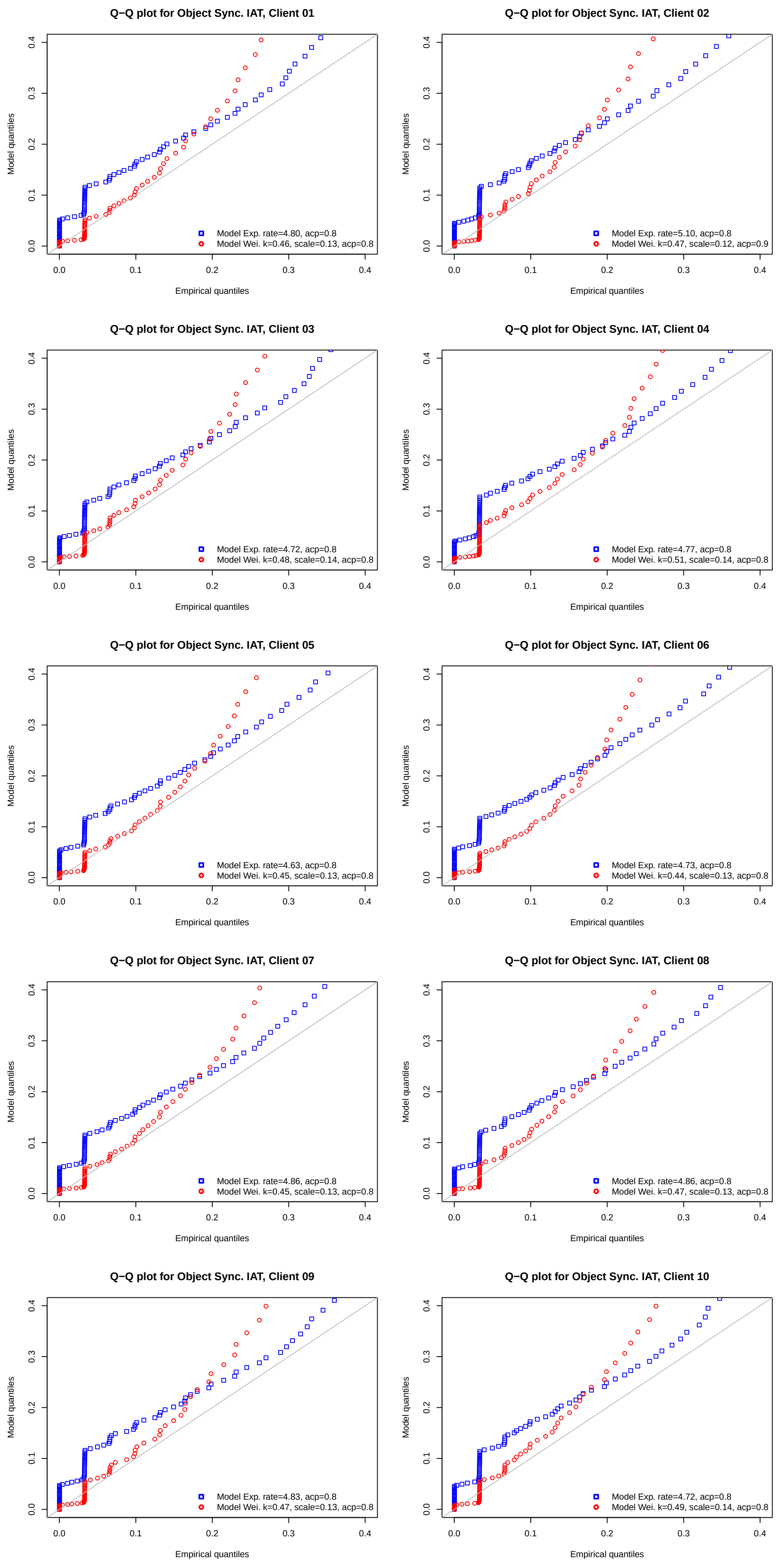


Fig. D.24.: Q-Q for OSIAT models, 10-player session, OWTB2

D.7 Pearson Correlation Coefficient for OSIAT models

The present section contains Tables D.8 and D.9 which show the Pearson correlation coefficient (PCC) values obtained for the OSIAT values from OWTB1 and OWTB2 respectively against the fitting distributions based on exponential and Weibull distributions proposed in the present work.

Tab. D.8.: Pearson's *r* for OSIAT exponential and Weibull-based models, OWTB1

Session	Client	r_{exp}	r_{wei}
2-client	1	0.997	0.997
	2	0.996	0.995
3-client	1	0.996	0.997
	2	0.998	0.998
	3	0.999	0.998
5-client	1	0.996	0.998
	2	0.996	0.996
	3	0.997	0.998
	4	0.985	0.966
	5	0.996	0.999

Tab. D.9.: Pearson's *r* for OSIAT exponential and Weibull-based models, OWTB2

Session	Client	r_{exp}	r_{wei}
1-client	1	0.96	0.99
2-client	1	0.97	0.99
	2	0.96	0.99
3-client	1	0.96	0.99
	2	0.96	0.99
	3	0.97	0.99
4-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.96	0.99
5-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.97	0.99
	5	0.96	0.99

Tab. D.9.: Pearson's r for OSIAT exponential and Weibull-based models, OWTB2

Session	Client	r_{exp}	r_{wei}
6-client	1	0.97	0.99
	2	0.97	0.99
	3	0.96	0.99
	4	0.96	0.99
	5	0.97	0.99
	6	0.97	0.99
7-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.96	0.99
	5	0.97	0.99
	6	0.97	0.99
	7	0.96	0.99
8-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.97	0.99
	5	0.98	0.99
	6	0.97	0.99
	7	0.97	0.99
	8	0.97	0.99
9-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.97	0.99
	5	0.98	0.99
	6	0.98	0.99
	7	0.97	0.99
	8	0.97	0.99
	9	0.97	0.99

Tab. D.9.: Pearson's r for OSIAT exponential and Weibull-based models, OWTB2

Session	Client	r_{exp}	r_{wei}
10-client	1	0.97	0.99
	2	0.97	0.99
	3	0.97	0.99
	4	0.96	0.99
	5	0.98	0.99
	6	0.98	0.99
	7	0.97	0.99
	8	0.97	0.99
	9	0.97	0.99
	10	0.97	0.99

D.8 Lambda Square values for OSIAT models

The present section contains Tables D.10 and D.11, which show the $\hat{\lambda}^2$ values obtained for OSIAT values measured in OWTB1 and OWTB2 against their respective fitting OSIAT models proposed in the present study.

Tab. D.10.: $\hat{\lambda}^2$ metric for exponential and Weibull-based models, OWTB1

Session	Client	$\hat{\lambda}^2_{exp}$	$\hat{\lambda}^2_{wei}$	$\sigma(\hat{\lambda}^2_{exp})$	$\sigma(\hat{\lambda}^2_{wei})$
2-client	1	14.33	14.26	837.45	824.49
	2	18.30	18.33	1347.37	1353.20
3-client	1	12.26	12.21	608.85	601.73
	2	4.37	4.38	86.77	88.16
	3	6.40	6.22	181.17	159.32
5-client	1	9.26	9.08	372.39	345.91
	2	7.74	7.78	265.54	271.29
	3	4.46	4.14	94.35	70.77
	4	23.14	24.70	2147.04	2778.04
	5	7.69	7.23	298.48	215.11

Tab. D.11.: $\hat{\lambda}^2$ metric for exponential and Weibull-based models, OWTB2

Session	Client	$\hat{\lambda}^2_{exp}$	$\hat{\lambda}^2_{wei}$	$\sigma(\hat{\lambda}^2_{exp})$	$\sigma(\hat{\lambda}^2_{wei})$
1-client	1	58.18	63.32	15347.74	20300.14
2-client	1	4.60	4.11	277.98	210.49
	2	3.26	2.60	324.23	200.11
3-client	1	2.81	2.09	148.40	75.77
	2	1.97	1.52	62.60	55.19
	3	1.60	1.17	23.56	12.98
4-client	1	6.45	5.85	255.86	184.93
	2	2.50	2.08	115.07	77.57
	3	1.87	1.35	64.23	37.05
	4	6.57	6.43	189.12	196.76
5-client	1	3.03	2.63	52.06	44.78
	2	1.85	1.20	265.23	130.62
	3	2.32	1.82	65.93	42.04
	4	5.83	5.56	149.43	138.94
	5	2.66	2.09	128.92	83.14
6-client	1	1.75	1.29	56.35	34.87
	2	1.94	1.58	34.78	26.49
	3	2.03	1.65	37.60	31.72
	4	8.39	7.82	606.05	462.42
	5	1.90	1.62	40.44	35.27
	6	2.67	2.32	44.80	39.64
7-client	1	2.48	1.83	115.82	56.51
	2	2.81	2.30	181.43	123.11
	3	1.70	1.33	41.34	33.29
	4	5.19	4.83	151.74	119.60
	5	2.03	1.63	34.19	23.35
	6	2.13	1.49	135.02	58.76
	7	2.16	1.59	107.65	61.14
8-client	1	31.02	31.25	3849.98	3920.16
	2	1.77	1.39	41.58	30.19
	3	2.10	1.64	52.43	30.66
	4	8.08	7.34	481.08	321.59
	5	1.87	1.62	27.95	27.10
	6	1.76	1.27	37.46	19.53
	7	2.91	2.17	312.36	156.68
	8	2.18	1.74	56.57	40.25

Tab. D.11.: $\hat{\lambda}^2$ metric for exponential and Weibull-based models, OWTB2

Session	Client	$\hat{\lambda}^2_{exp}$	$\hat{\lambda}^2_{wei}$	$\sigma(\hat{\lambda}^2_{exp})$	$\sigma(\hat{\lambda}^2_{wei})$
9-client	1	2.24	1.76	65.01	42.46
	2	2.13	1.69	74.74	44.85
	3	2.04	1.57	83.20	69.28
	4	7.61	6.59	848.72	469.75
	5	1.57	1.20	26.55	17.29
	6	2.03	1.50	91.29	45.57
	7	2.24	1.58	121.07	57.23
	8	1.98	1.55	29.57	19.96
	9	2.08	1.63	41.93	30.16
10-client	1	1.95	1.65	44.68	35.18
	2	2.65	2.12	184.13	131.39
	3	2.40	1.79	75.70	39.94
	4	5.01	4.68	132.71	108.29
	5	2.01	1.56	73.03	43.77
	6	2.07	1.63	53.49	30.45
	7	2.29	1.69	72.73	34.34
	8	1.83	1.31	41.29	21.18
	9	2.23	1.55	129.33	53.71
	10	2.85	2.15	184.36	91.41

D.9 Autocorrelation functions for OSPS

The present section contains a selection of ACF calculated for the object synchronisation packet size (OSPS) values from both OWTB1 and OWTB2.

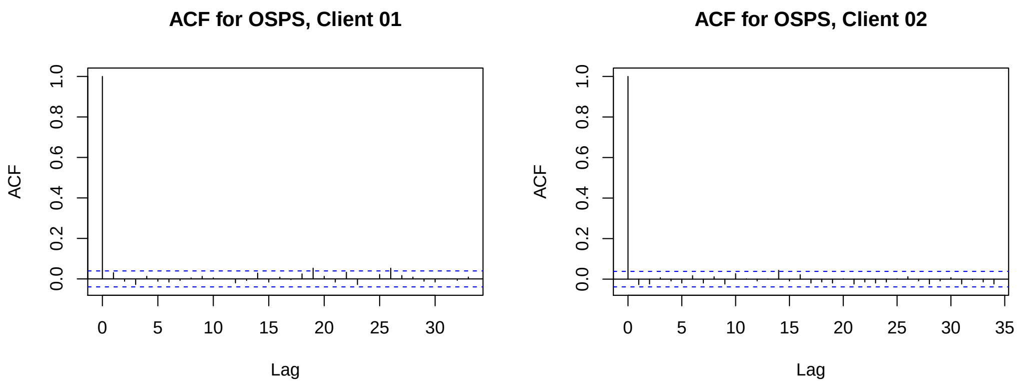


Fig. D.25.: ACF for OSPS, 2-player session, OWTB1

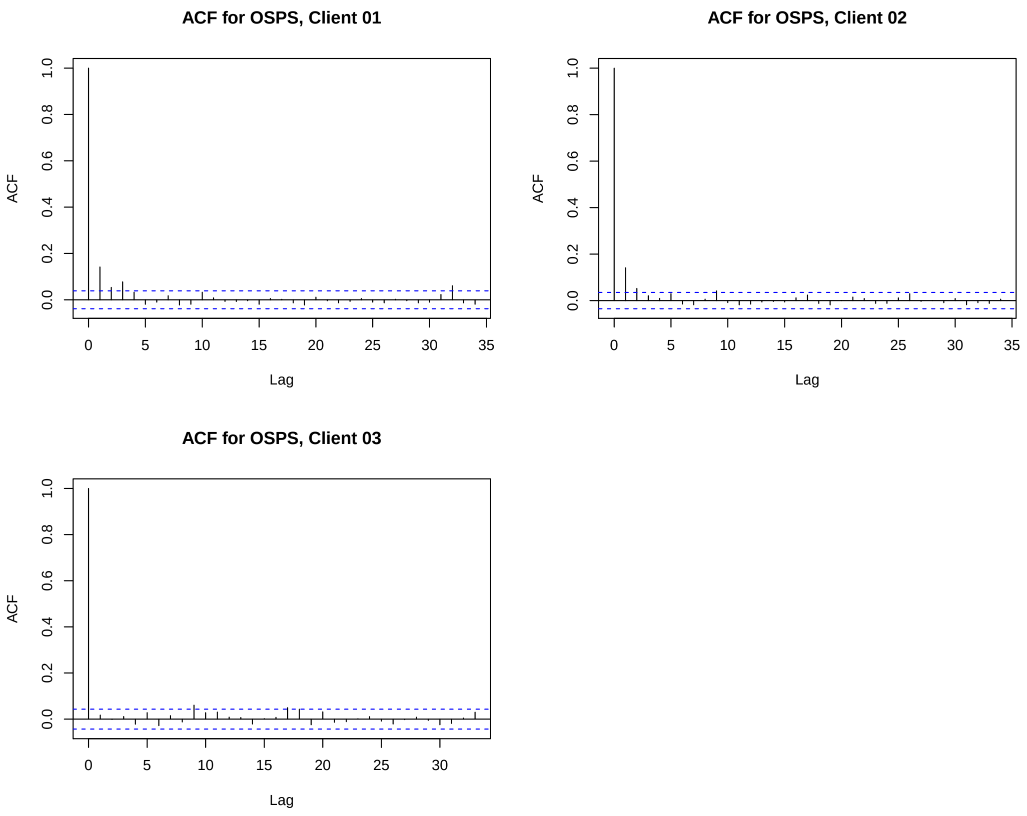


Fig. D.26.: ACF for OSPS, 3-player session, OWTB1

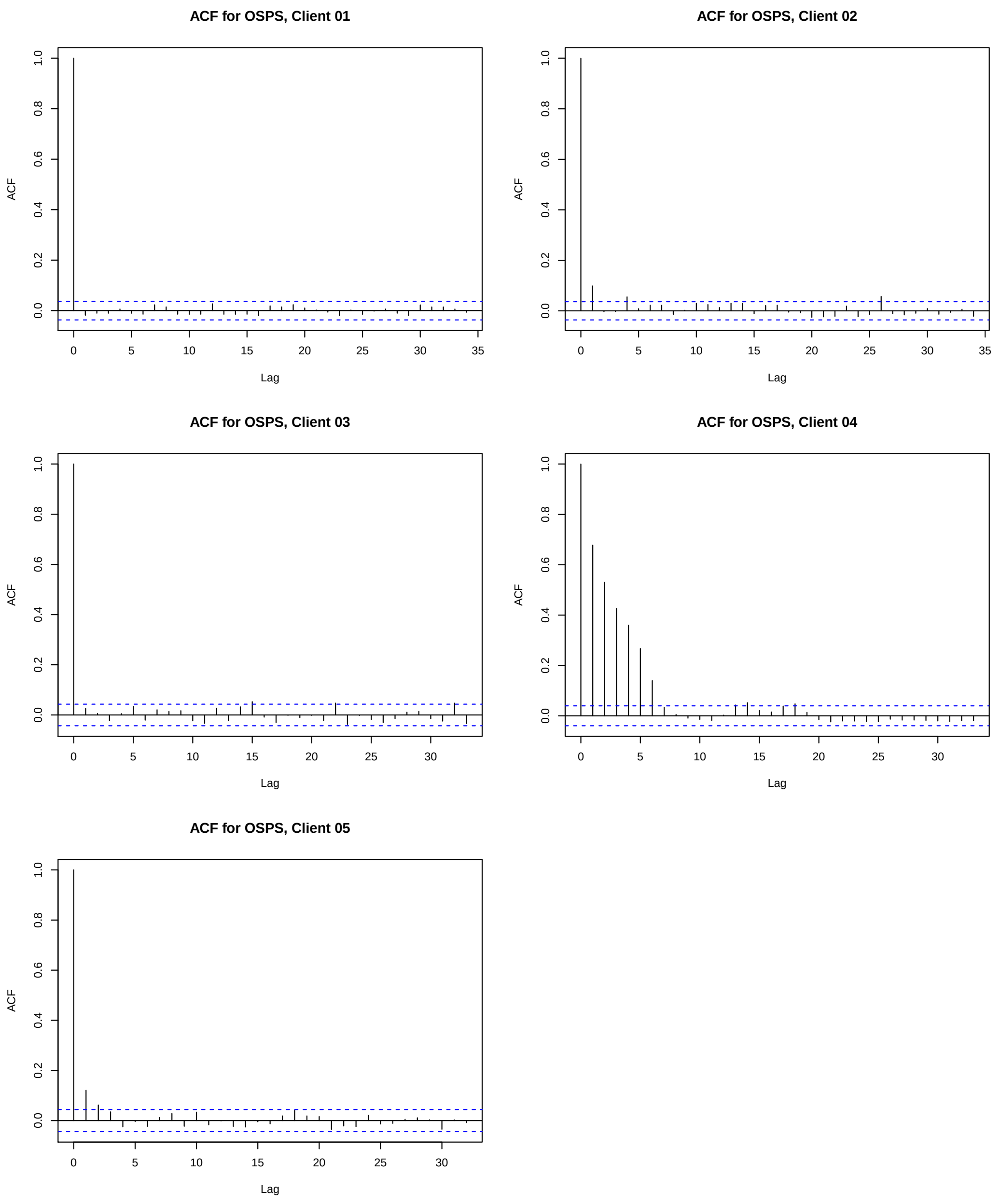


Fig. D.27.: ACF for OSPS, 5-player session, OWTB1

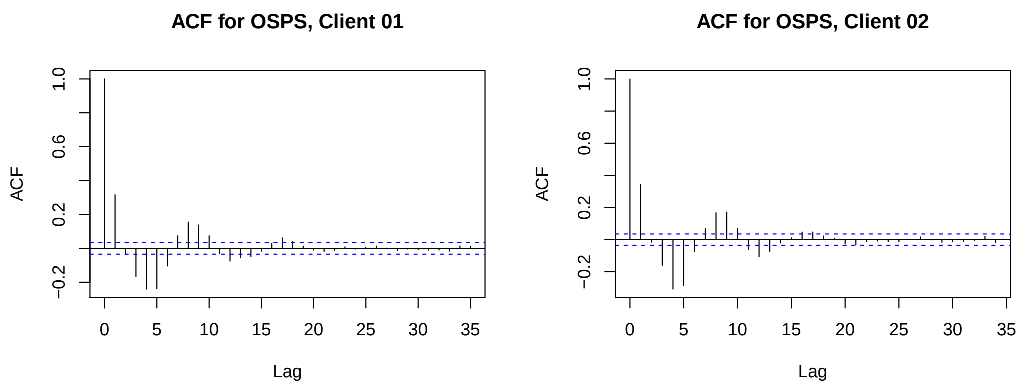


Fig. D.28.: ACF for OSPS, 2-player session, OWTB2

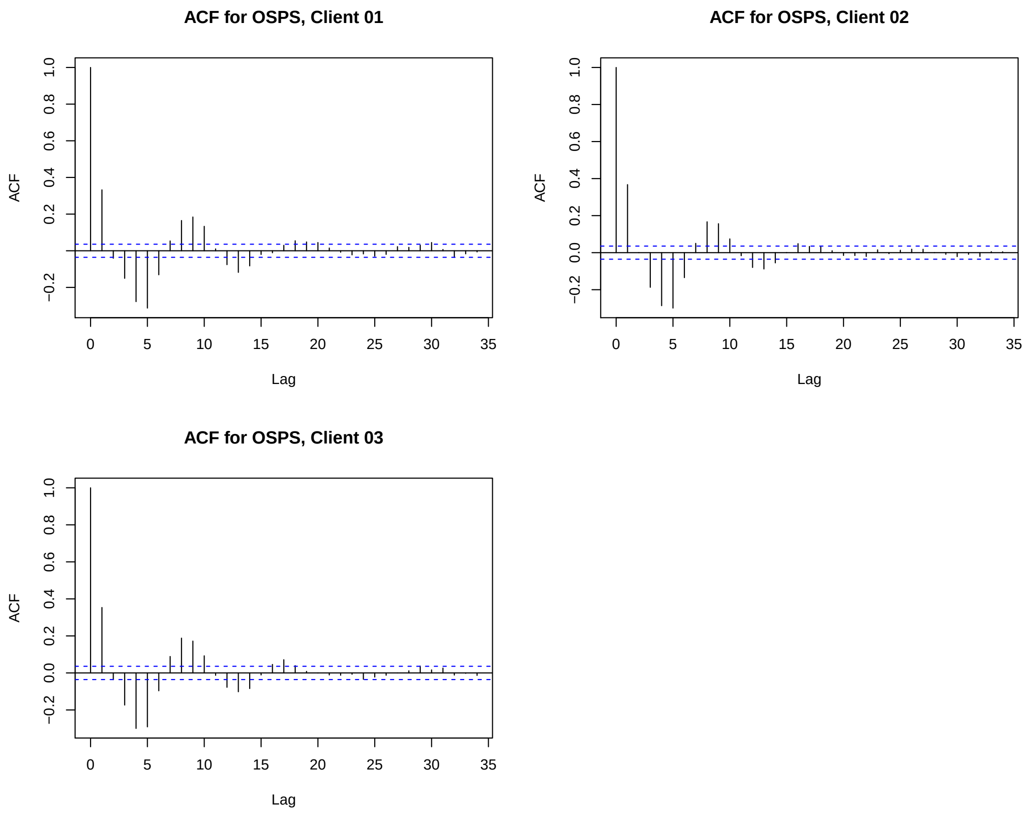


Fig. D.29.: ACF for OSPS, 3-player session, OWTB2

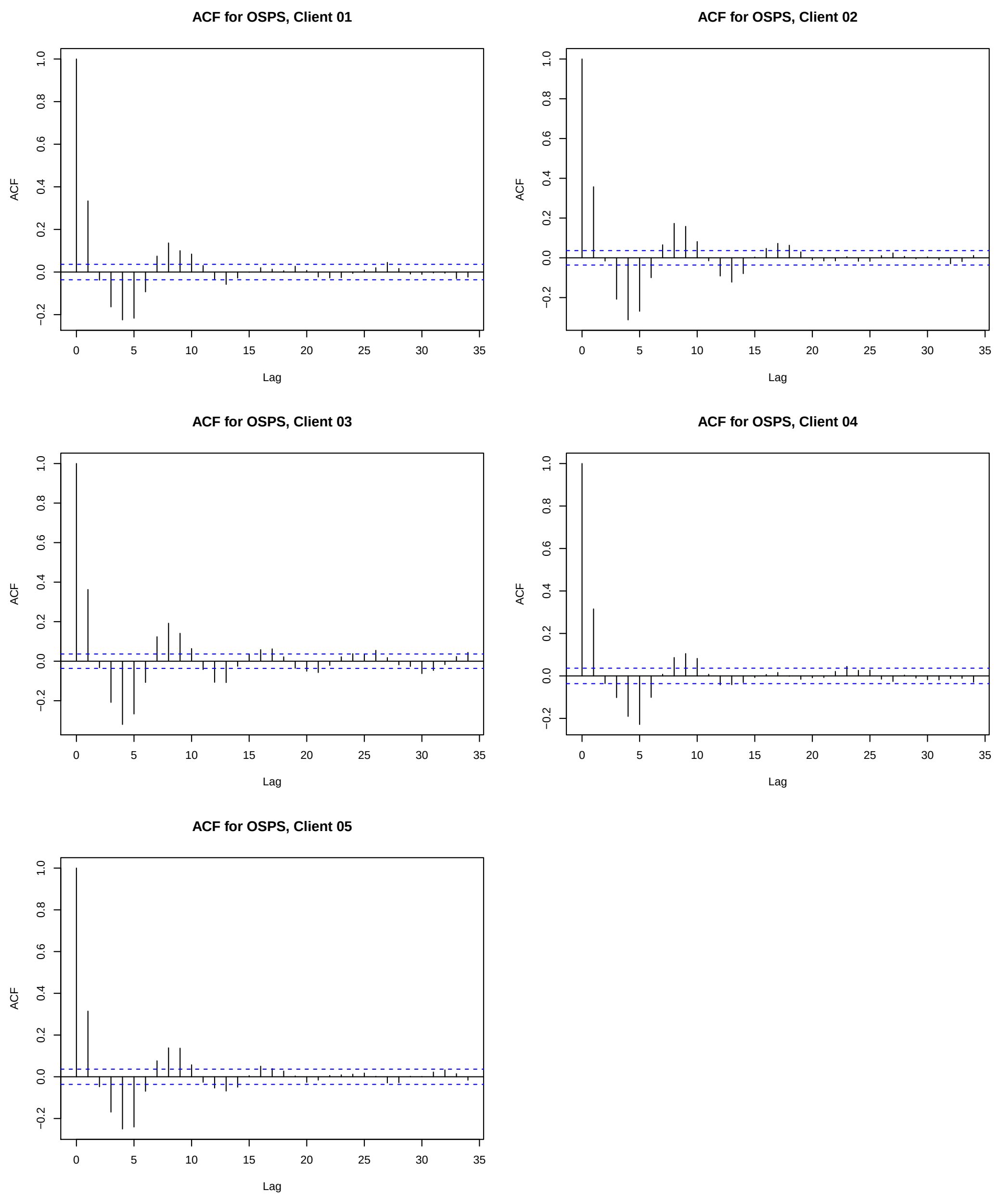


Fig. D.30.: ACF for OSPS, 5-player session, OWTB2

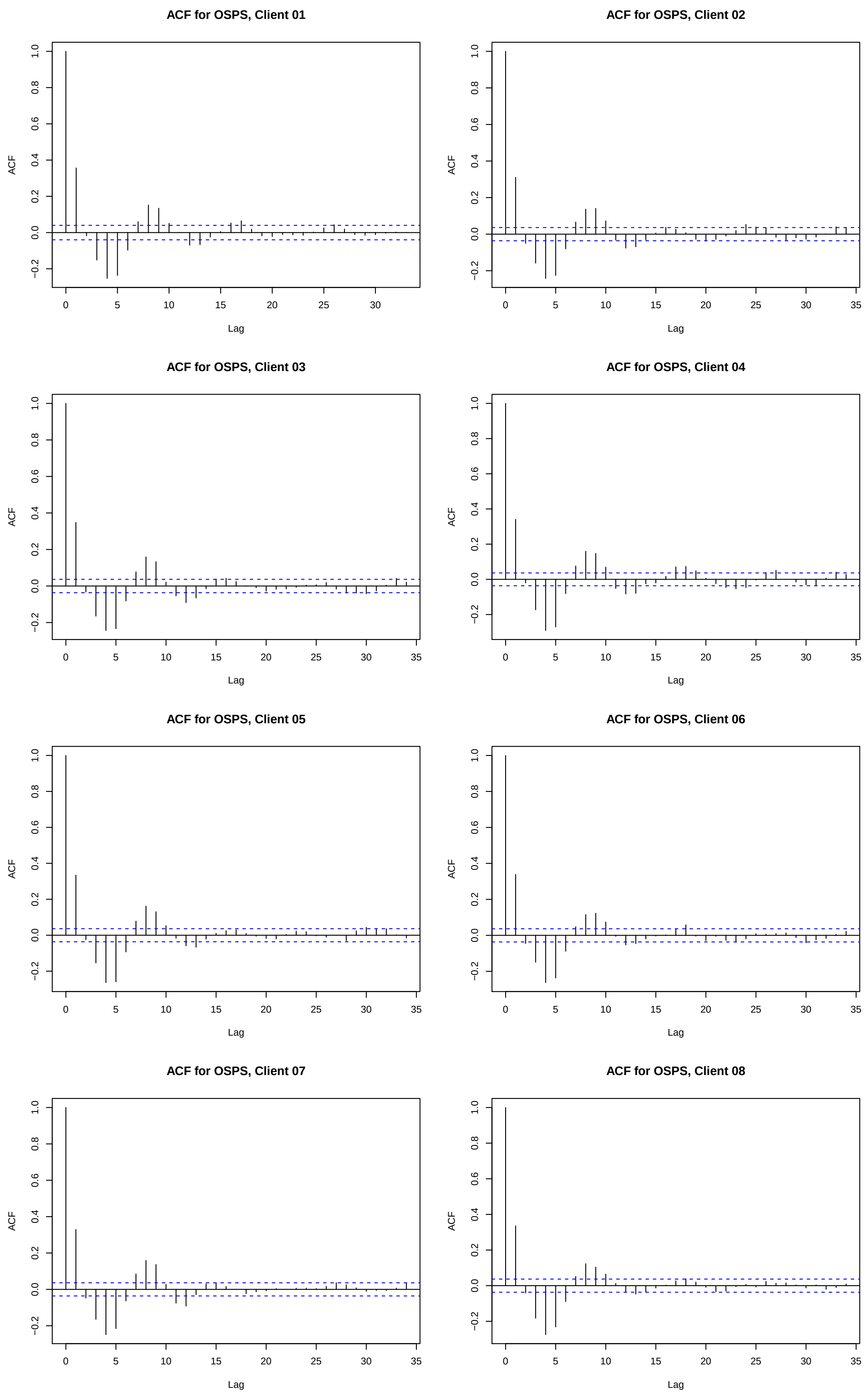


Fig. D.31.: ACF for OSPS, 8-player session, OWTB2

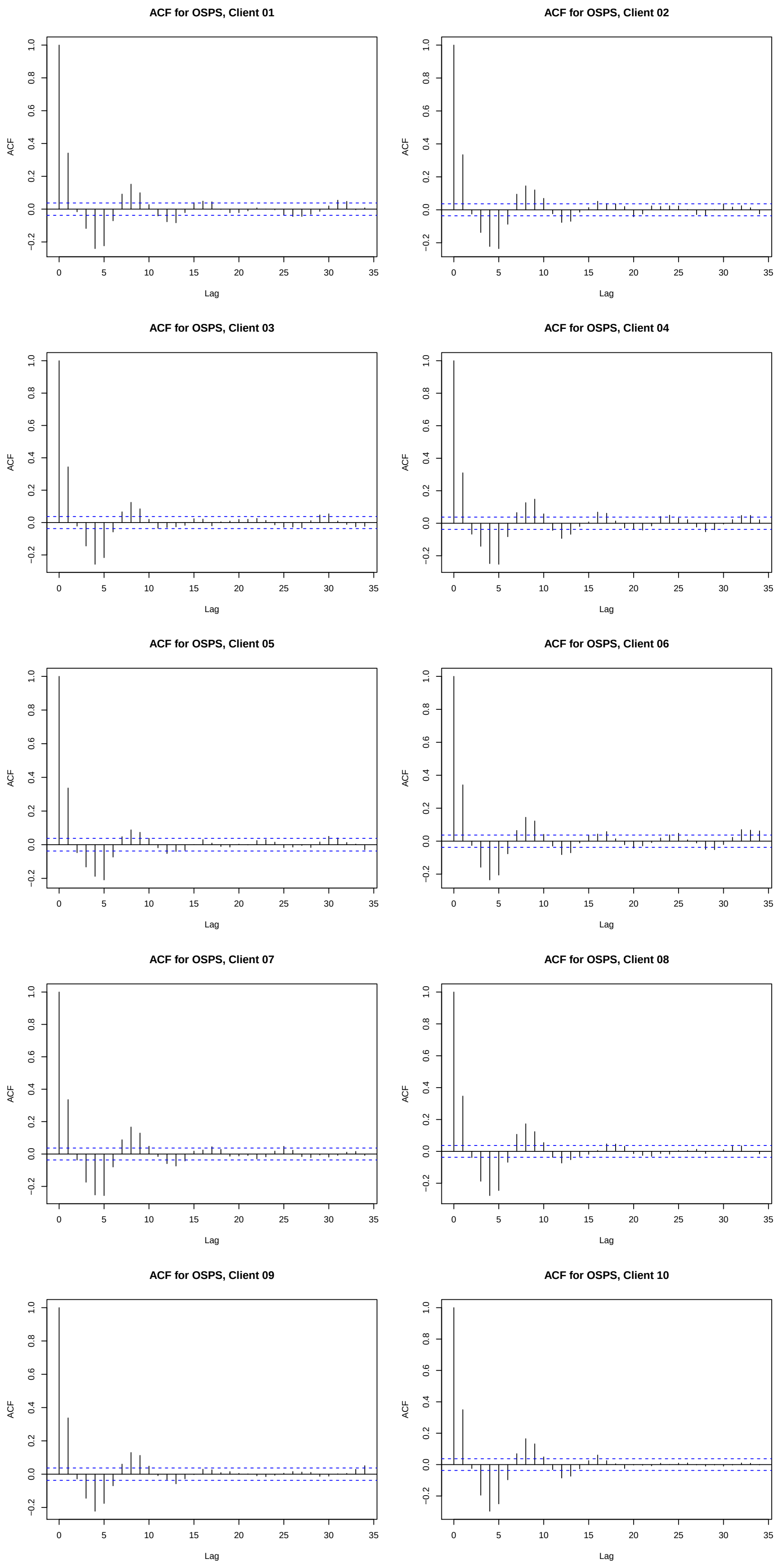


Fig. D.32.: ACF for OSPS, 10-player session, OWTB2

Bibliography

[Ach+16] Ugochi Acholonu, Jessa Dickinson, Leslie Smith, Dominic Amato, and Nichole Pinkard. „Understanding blended mentorship in Minecraft: Scaling computer science expertise across distances“. In: *2016 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. Atlanta, GA, USA: IEEE, Aug. 2016, pp. 1–1 (cit. on p. 12).

[Ada10] Ernest Adams. *Fundamentals of Game Design*. New Riders, Apr. 2010 (cit. on p. 4).

[Add] The CRPG Addict. *Moria (1975)*. `http://crpgaddict.blogspot.nl/2013/11/game-121-moria-1975.html`. (visited on 2015-10-24) (cit. on p. 5).

[ADS17] Carlos Arango, Rémy Dernat, and John Sanabria. „Performance Evaluation of Container-based Virtualization for High Performance Computing Environments“. In: *arXiv:1709.10140 [cs]* (Sept. 2017). arXiv: 1709.10140 (cit. on p. 106).

[Age] European Space Agency. *Home | Tropomi*. `http://www.tropomi.eu/`. (visited on 2019-07-14) (cit. on p. 14).

[Agu+08] C. Aguado Sanchez, J. Bloomer, P. Buncic, et al. „CVMFS - a file system for the CernVM virtual appliance“. In: *Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research*. Ed. by Monique Werlen Thomas Speer Federico Carminati. Nov. 2008, p. 52 (cit. on p. 105).

[Ald97] John Aldrich. „R.A. Fisher and the making of maximum likelihood 1912-1922“. In: *Statistical Science* 12.3 (Sept. 1997), pp. 162–176 (cit. on p. 47).

[Als+15a] T. Alstad, J.R. Duncan, S. Detlor, et al. „Minecraft computer game performance analysis and network traffic emulation by a custom bot“. In: *Science and Information Conference (SAI), 2015*. July 2015, pp. 227–236 (cit. on p. 12).

[Als+15b] T. Alstad, J. Riley Dunkin, S. Detlor, et al. „Game network traffic simulation by a custom bot“. In: *Systems Conference (SysCon), 2015 9th Annual IEEE International*. Apr. 2015, pp. 675–680 (cit. on p. 31).

[Ami+13] Rahul Amin, France Jackson, Juan E. Gilbert, Jim Martin, and Terry Shaw. „Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2“. In: *Human-Computer Interaction. Users and Contexts of Use*. Ed. by Masaaki Kurosu. Lecture Notes in Computer Science 8006. Springer Berlin Heidelberg, 2013, pp. 97–106 (cit. on pp. 8, 28).

[And+08] P. Andreetto, S. Andreozzi, G. Avellino, et al. „The gLite workload management system“. In: *Journal of Physics: Conference Series* 119.6 (July 2008), p. 062007 (cit. on p. 104).

[Arm] US Army. *America's Army*. `http://www.goarmy.com/downloads/americas-army-game.html`. (visited on 2015-09-26) (cit. on pp. 2, 3).

[Ars09] Dominic Arsenault. „Video Game Genre, Evolution and Innovation“. en-US. In: *Eludamos. Journal for Computer Game Culture* 3.2 (Oct. 2009), pp. 149–176 (cit. on p. 4).

[Aza05] Max Azarov. *Worst-case Ethernet Network Latency*. Tech. rep. 802.3 ResE study group, 2005, p. 4 (cit. on p. 28).

[BA06] P. Branch and G. Armitage. „Extrapolating server to client IP traffic from empirical measurements of first person shooter games“. In: *NetGames' 06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. Vol. 24. 2006 (cit. on pp. 29, 39, 49).

[Bai04] William Sims Bainbridge. „Berkshire Encyclopedia of Human-Computer Interaction“. In: *Berkshire Encyclopedia of Human-Computer Interaction*. Great Barrington, Mass: Berkshire Publishing Group, Oct. 2004, p. 474 (cit. on p. 7).

[Bal07] Ran Balicer. „Modeling Infectious Diseases Dissemination Through Online Role-Playing Games“. In: *Epidemiology* 18.2 (Mar. 2007), pp. 260–261 (cit. on p. 8).

[BAS13] Eliya Buyukkaya, Maha Abdallah, and Gwendal Simon. „A survey of peer-to-peer overlay approaches for networked virtual environments“. In: *Peer-to-Peer Networking and Applications* 8.2 (Sept. 2013), pp. 276–300 (cit. on pp. 10, 18, 21).

[Bay12] Jessica D. Bayliss. „Teaching game AI through Minecraft mods“. In: *2012 IEEE International Games Innovation Conference*. Rochester, NY, USA: IEEE, Sept. 2012, pp. 1–4 (cit. on p. 12).

[BB13] Gergo Balogh and Arpad Beszedes. „CodeMetrpolis &#x2014; A minecraft based collaboration tool for developers“. In: *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*. Eindhoven, Netherlands: IEEE, Sept. 2013, pp. 1–4 (cit. on p. 12).

[BD00] R.A. Bangun and E. Dutkiewicz. „Modelling multi-player games traffic“. In: *International Conference on Information Technology: Coding and Computing, 2000. Proceedings*. 2000, pp. 228–233 (cit. on pp. 36, 39, 47).

[BDA99] R.A. Bangun, E. Dutkiewicz, and G.J. Anido. „An analysis of multi-player network games traffic“. In: *1999 IEEE 3rd Workshop on Multimedia Signal Processing*. 1999, pp. 3–8 (cit. on p. 36).

[Bea+09] Leslie Beard, Kumanan Wilson, Dante Morra, and Jennifer Keelan. „A Survey of Health-Related Activities on Second Life“. In: *Journal of Medical Internet Research* 11.2 (May 2009) (cit. on p. 11).

[Bei+04] Tom Beigbeder, Rory Coughlan, Corey Lusher, et al. „The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®“. In: *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*. NetGames '04. New York, NY, USA: ACM, 2004, pp. 144–151 (cit. on pp. 12, 33).

[Ber+95] Jan Beran, Robert Sherman, M.S. Taqqu, and Walter Willinger. „Long-Range Dependence in Variable Bit Rate Video Traffic“. In: *Communications, IEEE Transactions on* 43 (Mar. 1995), pp. 1566–1579 (cit. on p. 36).

[Ber14] D. Bernstein. „Containers and Cloud: From LXC to Docker to Kubernetes“. In: *IEEE Cloud Computing* 1.3 (Sept. 2014), pp. 81–84 (cit. on p. 106).

[BHW07] Maged N. Kamel Boulos, Lee Hetherington, and Steve Wheeler. „Second Life: an overview of the potential of 3-D virtual worlds in medical and health education“. In: *Health Information & Libraries Journal* 24.4 (2007), pp. 233–245 (cit. on p. 11).

[Bor00] Michael S. Borella. „Source Models of Network Game Traffic“. In: *Computer Communications* 23.4 (Feb. 2000), pp. 403–410 (cit. on pp. 12, 13, 36, 39, 45, 47–49, 51, 60).

[BPS06] Ashwin R. Bharambe, Jeffrey Pang, and Srinivasan Seshan. „Colyseus: A Distributed Architecture for Online Multiplayer Games.“ In: *NSDI*. Vol. 6. 2006, pp. 12–12 (cit. on p. 10).

[Bra06] Philip Branch. „Measuring the autocorrelation of server to client traffic in first person shooter games“. In: *in Australian Telecommunications, Network and Applications Conference ATNAC*. 2006 (cit. on pp. 41, 42).

[BRH72] Ralph H. Baer, William T. Rusch, and William L. Harrison. „TELEVISION GAMING APPARATUS AND METHOD“. 3659285. Apr. 1972 (cit. on p. 4).

[Bru+13] Kjell Brunnström, Sergio Ariel Beker, Katrien De Moor, et al. *Qualinet white paper on definitions of quality of experience*. Tech. rep. Output from the 5^{th} Qualinet meeting, ref: *hal-00977812f*. Qualinet, Mar. 2013 (cit. on pp. 13, 19).

[BS06] K. O. Bowman and L. R. Shenton. „Estimation: Method of Moments“. In: *Encyclopedia of Statistical Sciences*. American Cancer Society, 2006 (cit. on p. 47).

[BV14] Newzoo BV. *PC Gaming Market to Total $24.4Bn in 2014*. `https://newzoo.com/insights/infographics/infographic-pcmmo-gaming-revenues-to-total-24-4bn-in-2014/`. (visited on 2019-06-22). 2014 (cit. on p. 4).

[BV18] Newzoo BV. *Newzoo Global Games Market Report 2018*. `https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2018-light-version/`. (visited on 2019-06-22). 2018 (cit. on p. 4).

[Çap99] Tolga K. Çapin, ed. *Avatars in networked virtual environments*. Chichester ; New York: Wiley, 1999 (cit. on p. 19).

[Cas+10] D. Cascado, S.J. Romero, S. Hors, et al. „Virtual worlds to enhance Ambient-Assisted Living“. In: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Sept. 2010, pp. 212–215 (cit. on p. 22).

[CB07] Anthony Cricenti and Philip Branch. „ARMA(1,1) modeling of Quake4 Server to client game traffic“. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*. NetGames ’07. New York, NY, USA: ACM, 2007, pp. 70–74 (cit. on pp. 8, 36, 41, 42, 48).

[CB97] Mark E. Crovella and Azer Bestavros. „Self-similarity in World Wide Web traffic: evidence and possible causes“. In: *IEEE/ACM Trans. Netw.* 5.6 (Dec. 1997), pp. 835–846 (cit. on p. 36).

[CBA07] Antonio L. Cricenti, Philip A. Branch, and Grenville J. Armitage. „Time-series modelling of server to client IP packet length in first person shooter games“. In: *Networks, 2007. ICON 2007. 15th IEEE International Conference on*. IEEE, 2007, pp. 507–512 (cit. on pp. 36, 41, 42).

[CC10] Mark Claypool and Kajal Claypool. „Latency can kill: precision and deadline in online games“. In: *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 215–222 (cit. on p. 33).

[CERa] CERN. *ALICE | CERN*. `https://home.cern/science/experiments/alice`. (visited on 2019-07-14) (cit. on p. 14).

[CERb] CERN. *ATLAS | CERN*. `https://home.cern/science/experiments/atlas`. (visited on 2019-07-14) (cit. on p. 14).

[CFM94] Ellaine Colburn, Steve Farrow, and Jim MsDonough. *ADST Multi-Service Distributed Training Testbed (MDT2). Lessons Learned.* Tech. rep. DTIC Document, 1994 (cit. on p. 2).

[Cha+83] John M. Chambers, William S. Cleveland, Paul A. Tukey, and Beat Kleiner. *Graphical Methods for Data Analysis*. Duxbury Press, 1983 (cit. on p. 48).

[Che+05] Kuan-Ta Chen, Polly Huang, Chun-Ying Huang, and Chin-Laung Lei. „Game Traffic Analysis: An MMORPG Perspective“. In: *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*. NOSSDAV ’05. New York, NY, USA: ACM, 2005, pp. 19–24 (cit. on p. 7).

[Che+06a] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. „An Empirical Evaluation of TCP Performance in Online Games“. In: *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*. ACE ’06. New York, NY, USA: ACM, 2006 (cit. on pp. 7, 36).

[Che+06b] Kuan-Ta Chen, Polly Huang, G.-S. Wang, Chun-Ying Huang, and Chin-Laung Lei. „On the Sensitivity of Online Game Playing Time to Network QoS“. In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. Apr. 2006, pp. 1–12 (cit. on p. 7).

[CHK17] Matt Cocar, Reneisha Harris, and Youry Khmelevsky. „Utilizing Minecraft bots to optimize game server performance and deployment“. In: *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. Windsor, ON: IEEE, Apr. 2017, pp. 1–5 (cit. on p. 12).

[Cho+12] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. „The brewing storm in cloud gaming: A measurement study on cloud to end-user latency“. In: *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE Press, 2012, p. 2 (cit. on p. 12).

[Chu+01] Elizabeth F. Churchill, David N. Snowdon, Alan J. Munro, Dan Diaper, and Colston Sanger, eds. *Collaborative Virtual Environments*. Computer Supported Cooperative Work. London: Springer London, 2001 (cit. on p. 1).

[Cla05] Mark Claypool. „The effect of latency on user performance in Real-Time Strategy games“. In: *Computer Networks*. Networking Issue in Entertainment Computing 49.1 (Sept. 2005), pp. 52–70 (cit. on p. 33).

[CLW03] M. Claypool, D. LaPoint, and J. Winslow. „Network analysis of counter-strike and starcraft“. In: *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*. 2003, pp. 261–268 (cit. on pp. 33, 95).

[Com] Open Wonderland Community. *Open Wonderland Community Wiki: Setting up an Open Wonderland Server Behind a NAT or Firewall*. `http://ow.cyramix.com/Wiki_page_Setting_up_an_Open_Wonderland_Server_Behind_a_NAT_or_Firewall.html`. (visited on 2019-05-20) (cit. on p. 56).

[Cor] Oracle Corporation. *GlassFish*. `https://javaee.github.io/glassfish/`. (visited on 2018-07-09) (cit. on p. 24).

[Dai+09] Dai Kai-yu, Lu Sheng-qi, Liu Gang, and Sun yi. „Research on path planning of intelligent virtual human in distributed virtual environment“. In: *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*. Shanghai, China: IEEE, Nov. 2009, pp. 899–903 (cit. on p. 21).

[Dan+01] Peter Danzig, Sugih Jamin, J Mitzel, and Deborah Estrin. „An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations“. In: *Int. Research and Exp.* 3 (June 2001) (cit. on p. 36).

[DGK75] Susan J. Devlin, R. Gnanadesikan, and J. R. Kettenring. „Robust Estimation and Outlier Detection with Correlation Coefficients“. In: *Biometrika* 62.3 (1975), pp. 531–545 (cit. on pp. 39, 75).

[Dis12] Working Group for Distributed Interactive Simulation. „IEEE Standard for Distributed Interactive Simulation–Application Protocols“. In: *IEEE Std 1278.1-2012 (Revision of IEEE Std 1278.1-1995)* (Dec. 2012), pp. 1–747 (cit. on p. 2).

[DPV05] Alberto Dainotti, Antonio Pescape, and Giorgio Ventre. „A packet-level Traffic Model of Starcraft“. In: *Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*. HOT-P2P ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 33–42 (cit. on pp. 36, 41, 48, 49).

[Ent10] Blizzard Entertainment. *Blizzard Entertainment:Press Releases*. *Link to Press Release*. (visited on 2019-06-22). Oct. 7, 2010 (cit. on p. 7).

[ES14] Herman A. Engelbrecht and Gregor Schiele. „Transforming Minecraft into a research platform“. In: *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. Las Vegas, NV: IEEE, Jan. 2014, pp. 257–262 (cit. on p. 12).

[Faj+15] E. M. Fajardo, J. M. Dost, B. Holzman, et al. „How much higher can HTCondor fly?“ In: *Journal of Physics: Conference Series* 664.6 (Dec. 2015), p. 062014 (cit. on p. 104).

[Fär02] J. Färber. „Network game traffic modelling“. In: *Proceedings of the 1st workshop on Network and system support for games*. 2002, pp. 53–57 (cit. on pp. 33, 36).

[Fär04] J. Färber. „Traffic modelling for fast action network games“. In: *Multimedia Tools and Applications* 23.1 (2004), pp. 31–46 (cit. on pp. 36, 39).

[FB14] Cynthia Foronda and Eric B. Bauman. „Strategies to Incorporate Virtual Simulation in Nurse Education“. In: *Clinical Simulation in Nursing* 10.8 (Aug. 2014), pp. 412–418 (cit. on p. 10).

[FCS13] J.L. Font, C. Callegari, and J.L. Sevillano-Ramos. „Experimental study and modelling of Networked Virtual Environment server traffic“. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. July 2013, pp. 1144–1149 (cit. on pp. 14, 22).

[Fen+02] W. Feng, F. Chang, W. Feng, and J. Walpole. „Provisioning on-line games: a traffic analysis of a busy counter-strike server“. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 2002, pp. 151–156 (cit. on pp. 8, 12, 33).

[Fen+05] Wu-chang Feng, F. Chang, Wu-chi Feng, and J. Walpole. „A traffic characterization of popular on-line games“. In: *IEEE/ACM Transactions on Networking* 13.3 (June 2005), pp. 488–500 (cit. on pp. 33, 36).

[Fer+07] Stênio Fernandes, Rafael Antonello, Josilene Moreira, Djamel Sadok, and Carlos Kamienski. „Traffic analysis beyond this world: the case of Second Life“. In: *17th International workshop on Network and operating systems support for digital audio and video, University of Illinois, Urbana-Champaign*. Citeseer, 2007, pp. 4–5 (cit. on p. 11).

[Fin13] Eric Finn. „Predicting the Perceived Quality of a First Person Shooter Game: the Team Fortress 2 T-Model“. PhD thesis. WORCESTER POLYTECHNIC INSTITUTE, 2013 (cit. on pp. 13, 19).

[Fis09] Paul A. Fishwick. „An Introduction to openSimulator and Virtual Environment Agent-based M&S Applications“. In: *Winter Simulation Conference*. WSC ’09. event-place: Austin, Texas. Winter Simulation Conference, 2009, pp. 177–183 (cit. on p. 10).

[Fis15] Ronald Aylmer Fisher. *Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population.* Journal article. 1915 (cit. on p. 75).

[FLK17] Artur Filipowicz, Jeremiah Liu, and Alain Kornhauser. „Learning to Recognize Distance to Stop Signs Using the Virtual World of Grand Theft Auto 5“. In: *Transportation Research Board* 96^{th} *Annual MeetingTransportation Research Board*. Jan. 2017 (cit. on p. 8).

[FM94] V.S. Frost and B. Melamed. „Traffic modeling for telecommunications networks“. In: *IEEE Communications Magazine* 32.3 (Mar. 1994), pp. 70–81 (cit. on p. 36).

[FO96] John L. Furlani and Peter W. Osel. „Abstract Yourself With Modules“. In: USENIX Association, Oct. 1996, pp. 193–204 (cit. on p. 105).

[Fog02] B. J. Fogg. *Persuasive Technology: Using Computers to Change What We Think and Do*. 1st ed. Morgan Kaufmann, Dec. 2002 (cit. on p. 22).

[Fon+10] Juan Luis Font, Pablo Iñigo, Manuel Domínguez, José Luis Sevillano, and Claudio Amaya. „Architecture, Design and Source Code Comparison of Ns-2 and Ns-3 Network Simulators“. In: *Proceedings of the 2010 Spring Simulation Multiconference,* 13^{th} *Communication & Networking Simulation Symposium*. SpringSim ’10. San Diego, CA, USA: Society for Computer Simulation International, 2010, 109:1–109:8 (cit. on pp. 14, 85, 86).

[Fon+11a] J. L. Font, D. Cascado, J. L. Sevillano, et al. „Network requirements evaluation of a multi-user virtual environment“. In: *2011 International Symposium on Performance Evaluation of Computer Telecommunication Systems*. June 2011, pp. 90–97 (cit. on pp. 13, 85, 86).

[Fon+11b] Juan Luis Font, Pablo Iñigo, Manuel Domínguez, José Luis Sevillano, and Claudio Amaya. „Analysis of source code metrics from ns-2 and ns-3 network simulators“. In: *Simulation Modelling Practice and Theory* 19.5 (May 2011), pp. 1330–1346 (cit. on p. 14).

[Fon+12a] J.L. Font, J.L. Sevillano, D. Cascado-Caballero, G. Lopez-Munoz, and B. Regassa. „Design, implementation and validation of a simulation tool for Networked Virtual Environments“. In: *2012 International Conference on Computer, Information and Telecommunication Systems (CITS)*. May 2012, pp. 1–5 (cit. on pp. 14, 22).

[Fon+12b] Juan L. Font, Daniel Cascado, José L. Sevillano, Fernando Díaz del Río, and Gabriel Jiménez. „Network traffic analysis and evaluation of a multi-user virtual environment“. In: *Simulation Modelling Practice and Theory* 26 (Aug. 2012), pp. 1–15 (cit. on pp. 13, 32, 33, 55).

[Foua] AutoHotKey Foundation. *AutoHotkey Site*. `http://www.autohotkey.com/`. (visited on 2012-12-04) (cit. on pp. 32, 108).

[Foub] AutoHotKey Foundation. *Random number generation in AutoHotKey*. `http://www.autohotkey.com/docs/commands/Random.htm`. (visited on 2012-12-04) (cit. on p. 32).

[Fouc] Open Wonderland Foundation. *jVoiceBridge*. *Link to jVoiceBridge documentation*. (visited on 2018-07-09) (cit. on p. 23).

[Foud] Open Wonderland Foundation. *MTGame*. *Link to MTGame Library*. (visited on 2018-07-09) (cit. on p. 23).

[Foue] Open Wonderland Foundation. *Open Wonderland Forum – Google Groups*. *Link to Open Wonderland Google Groups*. (visited on 2018-07-09) (cit. on p. 57).

[Fouf] Open Wonderland Foundation. *Openwonderland Architecture*. *Link to Open Wonderland architecture documentation*. (visited on 2018-07-09) (cit. on pp. 25, 56, 62).

[Foug] Python Software Foundation. *Python.org*. `https://www.python.org/about/`. (visited on 2019-06-06) (cit. on p. 35).

[Fouh] R Foundation. *R: What is R?* `https://www.r-project.org/about.html`. (visited on 2019-06-06) (cit. on p. 35).

[FRS05] Tobias Fritsch, Hartmut Ritter, and Jochen Schiller. „The effect of latency and network limitations on MMORPGs: a field study of everquest2“. In: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*. NetGames ’05. New York, NY, USA: ACM, 2005, pp. 1–9 (cit. on p. 7).

[FSC12] J.L. Font, J.L. Sevillano, and D. Cascado. „An experimental evaluation of server performance in Networked Virtual Environments“. In: *2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. July 2012, pp. 1–6 (cit. on pp. 22, 27).

[Gar+09] M. Gardner, G. A. Gánem-Gutiérrez, J. Scott, and C. Fowler. *Designing and Building immersive education spaces using Project Wonderland: from pedagogy through to practice*. Other. 2009 (cit. on p. 22).

[Gar+11] Michael Gardner, Gabriela Adela Gánem-Gutiérrez, John Scott, Bernard Horan, and Victor Callaghan. *Immersive Education Spaces Using Open Wonderland: From Pedagogy through to Practice*. Book Section. 2011 (cit. on p. 22).

[Gar89] Richard E. Jr Garvey. „SIMNET-D: Extending Simulation Boundaries“. In: *Journal of the American Defense Preparedness Association* (1989) (cit. on p. 2).

[GB95] Chris Greenhalgh and Steve Benford. „MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading“. In: *ICDCS*. 1995 (cit. on p. 9).

[GB96] Matthias Grossglauser and Jean-Chrysostome Bolot. *On the Relevance of Long-Range Dependence in Network Traffic*. report. Mar. 1996 (cit. on p. 36).

[Geb+14] Sascha Gebhardt, Sebastian Pick, Thomas Oster, Bernd Hentschel, and Torsten Kuhlen. „An evaluation of a smart-phone-based menu system for immersive virtual environments“. In: *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. MN, USA: IEEE, Mar. 2014, pp. 31–34 (cit. on p. 20).

[Get+10] K. Getchell, I. Oliver, A. Miller, and C. Allison. „Metaverses as a Platform for Game Based Learning“. In: *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Apr. 2010, pp. 1195–1202 (cit. on p. 11).

[GG15] C. Gittens and J. Greaves. „Transforming BrowserQuest into an epidemiological tool for modelling disease dissemination“. In: *2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*. July 2015, pp. 143–148 (cit. on pp. 8, 10).

[GHT08] Judith Good, Katherine Howland, and Liz Thackray. „Problem-based learning spanning real and virtual words: a case study in Second Life“. In: *Research in Learning Technology* 16.3 (Sept. 2008) (cit. on p. 11).

[GS10] Leif Gustafsson and Mikael Sternad. „Consistent micro, macro and state-based population modelling“. In: *Mathematical Biosciences* 225.2 (June 2010), pp. 94–107 (cit. on p. 35).

[Gun+17] Simon Gunkel, Martin Prins, Hans Stokking, and Omar Niamut. „WebVR meets WebRTC: Towards 360-degree social VR experiences“. In: *2017 IEEE Virtual Reality (VR)*. Los Angeles, CA, USA: IEEE, 2017, pp. 457–458 (cit. on pp. 12, 20).

[Hag96] O. Hagsand. „Interactive multiuser VEs in the DIVE system“. In: *IEEE MultiMedia* 3.1 (1996), pp. 30–39 (cit. on p. 9).

[Hay06] Elisabeth Hayes. „Situated Learning in Virtual Worlds: The Learning Ecology of Second Life“. In: *Adult Education Research Conference* (July 2006) (cit. on p. 11).

[Haz95] Michiel Hazewinkel, ed. *Encyclopaedia of Mathematics (set)*. 1st ed. Springer, July 1995 (cit. on p. 40).

[HB01] Tristan Henderson and Saleem Bhatti. „Modelling user behaviour in networked games“. In: *Proceedings of the ninth ACM international conference on Multimedia*. MULTIMEDIA '01. New York, NY, USA: ACM, 2001, pp. 212–220 (cit. on p. 36).

[He+17] Tianyu He, Xiaoming Chen, Zhibo Chen, et al. „Immersive and collaborative Taichi motion learning in various VR environments“. In: *2017 IEEE Virtual Reality (VR)*. Los Angeles, CA, USA: IEEE, 2017, pp. 307–308 (cit. on p. 20).

[Hen14] Tom Henderson. *Ns-3 Overview*. `https://www.nsnam.org/docs/ns-3-overview.pdf`. (visited on 2019-05-06). July 2014 (cit. on p. 86).

[HL04] Shun-Yun Hu and Guan-Ming Liao. „Scalable Peer-to-peer Networked Virtual Environment“. In: *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*. NetGames '04. New York, NY, USA: ACM, 2004, pp. 129–133 (cit. on p. 21).

[Hos90] J. R. M. Hosking. „L-Moments: Analysis and Estimation of Distributions Using Linear Combinations of Order Statistics“. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 52.1 (1990), pp. 105–124 (cit. on p. 47).

[HS17] Juho Hamari and Max Sjöblom. „What is eSports and why do people watch it?“ In: *Internet Research* 27.2 (Mar. 2017), pp. 211–232 (cit. on p. 8).

[Hüb08] Christian Hübsch. „Analyzing Unreal Tournament 2004 Network Traffic Characteristics“. In: *Computer Games & Allied Technology 08 (CGAT 08): animation, multimedia, IPTV & edutainment: proceedings, Singapore*. Ed. by Edmond Prakash. Apr. 2008 (cit. on p. 36).

[Int] Bohemian Interactive. *New Developments in VBS3*. *Link to document*. (visited on 2015-09-26). Orlando, Florida (cit. on p. 3).

[Int12] Zipper Interactive. *Zipper Interactive: Official Website*. `https://web.archive.org/web/20120424114812/http://zipperint.com/`. (visited on 2015-09-26). Apr. 2012 (cit. on p. 3).

[JEP10] J. A. Joergensen, L. Ellekilde, and H. G. Petersen. „RobWorkSim - an Open Simulator for Sensor based Grasping“. In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. June 2010, pp. 1–8 (cit. on p. 10).

[KC08] James Kinicki and Mark Claypool. „Traffic Analysis of Avatars in Second Life“. In: *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. NOSSDAV '08. New York, NY, USA: ACM, 2008, pp. 69–74 (cit. on p. 11).

[Kim+05] Jaecheol Kim, Jaeyoung Choi, Dukhyun Chang, et al. „Traffic Characteristics of a Massively Multi-player Online Role Playing Game“. In: *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*. NetGames '05. New York, NY, USA: ACM, 2005, pp. 1–8 (cit. on p. 7).

[Kle75] Leonard Kleinrock. *Queueing Systems. Volume 1: Theory*. 1 edition. New York: Wiley-Interscience, Jan. 1975 (cit. on p. 36).

[KSB17] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. „Singularity: Scientific containers for mobility of compute“. In: *PLOS ONE* 12.5 (May 2017), e0177459 (cit. on p. 106).

[Kus04] David Kushner. „Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture“. In: *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture*. Edición: Reprint. New York: Random House Inc, May 2004, pp. 182–187 (cit. on p. 7).

[KY11] J. Kaplan and N. Yankelovich. „Open Wonderland: An Extensible Virtual World Architecture“. In: *IEEE Internet Computing* 15.5 (Sept. 2011), pp. 38–45 (cit. on p. 22).

[LA03] T. Lang and G. Armitage. „A ns2 model for the Xbox system link game HALO“. In: *traffic* 1 (2003), p. 3 (cit. on p. 86).

[Lab13] Linden Lab. *Infographic: 10 years of Second Life*. `https://www.lindenlab.com/releases/infographic-10-years-of-second-life`. (visited on 2019-06-05). June 20, 2013 (cit. on p. 11).

[Lan+03] T. Lang, G. Armitage, P. Branch, and H. Y. Choo. „A synthetic traffic model for Half-Life“. In: *Australian Telecommunications Networks & Applications Conference*. Vol. 2003. 2003 (cit. on p. 36).

[LB10] Huaiyu Liu and M. Bowman. „Scale Virtual Worlds through Dynamic Load Balancing“. English. In: *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, Oct. 2010, pp. 43–52 (cit. on p. 11).

[LBA04] T. Lang, P. Branch, and G. Armitage. „A synthetic traffic model for Quake3“. In: *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 2004, pp. 233–238 (cit. on p. 36).

[LC17] S. W. K. Lee and R. K. C. Chang. „On "shot around a corner" in first-person shooter games“. In: *2017 15th Annual Workshop on Network and Systems Support for Games (NetGames)*. June 2017, pp. 1–6 (cit. on p. 8).

[Le 79] L. M. Le Cam. *Maximum likelihood: an introduction*. Statistics Branch, Department of Mathematics, University of Maryland, 1979 (cit. on p. 47).

[Lel+93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. „On the self-similar nature of Ethernet traffic“. In: *ACM SIGCOMM Computer Communication Review*. Vol. 23. ACM, 1993, pp. 183–193 (cit. on p. 36).

[LH13] Charles J. Lesko and Yolanda A. Hollingsworth. „Architecting Scalable Academic Virtual World Grids: A Case Utilizing OpenSimulator“. In: *Journal For Virtual Worlds Research* 6.1 (Apr. 2013) (cit. on pp. 10, 103).

[LHR02] Jani Lakkakorpi, Andreas Heiner, and Jussi Ruutu. *Measurement and characterization of Internet gaming traffic*. Tech. rep. P.O. Box 407, FIN-00045 NOKIA GROUP, Finland: Nokia Research Center, Jan. 2002 (cit. on pp. 36, 49).

[Lia+09] Huiguang Liang, Ransi Nilaksha De Silva, Wei Tsang Ooi, and Mehul Motani. „Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications“. In: *Multimedia Tools and Applications* 45.1-3 (May 2009), pp. 163–190 (cit. on p. 11).

[Lui01] J.C.S. Lui. „Constructing communication subgraphs and deriving an optimal synchronization interval for distributed virtual environment systems“. In: *IEEE Transactions on Knowledge and Data Engineering* 13.5 (2001), pp. 778–792 (cit. on p. 56).

[Mac87] M. H. (Myron H.) MacDougall. *Simulating computer systems : techniques and tools*. MIT Press, Cambridge, Mass., 1987 (cit. on p. 34).

[MAK] VT MAK. *VT MAK Official Site*. `http://www.mak.com/products.html`. (visited on 2015-09-26) (cit. on p. 3).

[Met] Inc. MetaVR. *MetaVR real-time PC-based 3D visual simulation*. `https://www.metavr.com/`. (visited on 2015-09-26) (cit. on p. 3).

[MF08] Chip Morningstar and F. Randall Farmer. „The Lessons of Lucasfilm's Habitat". In: *Journal For Virtual Worlds Research* 1.1 (2008) (cit. on p. 6).

[MF91] Chip Morningstar and F. Randall Farmer. „Cyberspace". In: ed. by Michael Benedikt. Cambridge, MA, USA: MIT Press, 1991, pp. 273–302 (cit. on p. 6).

[MH12] Matthew T. Marino and Michael T. Hayes. „Promoting inclusive education, civic scientific literacy, and global citizenship with videogames". In: *Cultural Studies of Science Education* 7.4 (Dec. 2012), pp. 945–954 (cit. on p. 8).

[Mob] MobyGames. *Drew Johnston Video Game Credits and Biography*. `http://www.mobygames.com/developer/sheet/view/developerId,277286/`. (visited on 2015-09-26) (cit. on p. 3).

[MP03] Jessica Mulligan and Bridgette Patrovsky. „Developing Online Games: An Insider's Guide". In: *Developing Online Games: An Insider's Guide*. Indianapolis, Ind: New Riders Games, Mar. 2003, p. 444 (cit. on p. 5).

[MSB13] S. Moller, S. Schmidt, and J. Beyer. „Gaming taxonomy: An overview of concepts and evaluation methods for computer gaming QoE". In: *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. July 2013, pp. 236–241 (cit. on pp. 4, 13).

[MT95] D.C. Miller and J.A. Thorpe. „SIMNET: the advent of simulator networking". In: *Proceedings of the IEEE* 83.8 (Aug. 1995), pp. 1114–1123 (cit. on p. 2).

[Myu03] In Jae Myung. „Tutorial on maximum likelihood estimation". In: *Journal of Mathematical Psychology* 47.1 (Feb. 2003), pp. 90–100 (cit. on p. 47).

[NC04] James Nichols and Mark Claypool. „The Effects of Latency on Online Madden NFL Football". In: *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. NOSSDAV '04. New York, NY, USA: ACM, 2004, pp. 146–151 (cit. on p. 28).

[NSR16] Steve Nebel, Sascha Schneider, and Günter Daniel Rey. „Mining Learning and Crafting Scientific Experiments: A Literature Review on the Use of Minecraft in Education and Research". In: *Educational Technology & Society* 19 (Jan. 2016), pp. 355–366 (cit. on pp. 11, 12).

[OB10] Mohammed S. Obaidat and Noureddine A. Boudriga. *Fundamentals of Performance Evaluation of Computer and Telecommunications Systems*. New York, NY, USA: Wiley-Interscience, 2010 (cit. on p. 85).

[Oos94] Roland Oosterbaan. „Frequency and regression analysis". In: *IILRI*. Vol. 16. Jan. 1994, pp. 175–224 (cit. on p. 47).

[Pax93] Vern Paxson. *Empirically-derived analytic models of wide-area TCP connections: Extended report*. Tech. rep. technical report LBL-34086, Lawrence Berkeley Laboratory, May, 1993. Available as WANTCP-models. 1. ps. Z and WAN-TCP-models. 2. ps. Z via anonymous FTP to ftp. ee. lbl. gov, 1993 (cit. on pp. 42, 49, 51).

[Pax94] V. Paxson. „Empirically derived analytic models of wide-area TCP connections". In: *IEEE/ACM Transactions on Networking* 2.4 (Aug. 1994), pp. 316–336 (cit. on p. 36).

[PF95] Vern Paxson and Sally Floyd. „Wide area traffic: the failure of Poisson modeling". In: *IEEE/ACM Transactions on Networking (ToN)* 3.3 (1995), pp. 226–244 (cit. on p. 36).

[PH94] Johann Pfanzagl and R. Hamböker. *Parametric Statistical Theory*. Walter de Gruyter, 1994 (cit. on p. 47).

[PJ90] Shane P. Pederson and Mark E. Johnson. „Estimating Model Discrepancy". In: *Technometrics* 32.3 (Aug. 1990), p. 305 (cit. on pp. 49–51).

[PKK05] HyoJoo Park, TaeYong Kim, and SaJoong Kim. „Network Traffic Analysis and Modeling for Games". In: *Internet and Network Economics*. Ed. by Xiaotie Deng and Yinyu Ye. Lecture Notes in Computer Science 3828. Springer Berlin Heidelberg, Dec. 2005, pp. 1056–1065 (cit. on pp. 7, 48).

[Proa] Ns-2 Project. *The Network Simulator - ns-2*. `https://www.isi.edu/nsnam/ns/`. (visited on 2019-06-02) (cit. on p. 86).

[Prob] Ns-3 Project. *Ns-3 project on Gitlab*. `https://gitlab.com/nsnam/ns-3-dev`. (visited on 2019-05-26) (cit. on p. 86).

[Proc] Open Wonderland Project. *Open Wonderland*. `http://www.openwonderland.org/`. (visited on 2011-03-15) (cit. on pp. 22, 59).

[Prod] Open Wonderland Project. *Wonderland Tutorial*. `http://code.google.com/p/openwonderland/wiki/OpenWonderland`. (visited on 2018-07-09) (cit. on p. 24).

[Proe] RedDwarf Project. *RedDwarf Project Documentation*. *Link to RedDwarf site*. (visited on 2011-12-30) (cit. on p. 24).

[Prof] Squid Project. *Squid : Optimising Web Delivery*. http://www.squid-cache.org/. (visited on 2019-06-27) (cit. on p. 105).

[Prog] Wireshark Project. *Wireshark Checksums*. `https://www.wireshark.org/docs/wsug_html_chunked/ChAdvChecksums.html`. (visited on 2019-07-10) (cit. on p. 33).

[Proh] XENON Project. *Homepage of the XENON1T Dark Matter Search*. `http://www.xenon1t.org/`. (visited on 2019-07-14) (cit. on p. 14).

[Pro19] Ns-3 Project. *Ns-3 Documentation*. `https://www.nsnam.org/documentation/`. (visited on 2019-04-06). Apr. 6, 2019 (cit. on pp. 85, 86, 92).

[Rab15] Steven Rabin, ed. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. A K Peters/CRC Press, Apr. 2015 (cit. on p. 11).

[Ran84] Bo Ranneby. „The Maximum Spacing Method. An Estimation Method Related to the Maximum Likelihood Method“. In: *Scandinavian Journal of Statistics* 11.2 (1984), pp. 93–112 (cit. on p. 47).

[RB06] Jason Rutter and Jo Bryce. *Understanding Digital Games*. SAGE, Apr. 2006 (cit. on p. 11).

[Rel07] US Army News Release. *"Virtual Army Experience" Traveling Exhibit Offers Hands-on Test Drive of Soldiering in U.S. Army*. *Link to press release*. (visited on 2015-09-26). July 2007 (cit. on p. 3).

[Rep+14] Alexander Repenning, David C. Webb, Catharine Brand, et al. „Beyond Minecraft: Facilitating Computational Thinking through Modeling and Programming in 3D“. In: *IEEE Computer Graphics and Applications* 34.3 (May 2014), pp. 68–71 (cit. on p. 12).

[RG14] Kjetil Raaen and Tor-Morten Grønli. „Latency Thresholds for Usability in Games: A Survey“. In: *Norsk Informatikkonferanse (NIK)* (2014) (cit. on p. 19).

[Rhe91] Howard Rheingold. *Virtual reality*. Summit Books, 1991 (cit. on p. 2).

[RHS10] S. Ratti, B. Hariri, and S. Shirmohammadi. „A Survey of First-Person Shooter Gaming Traffic on the Internet“. In: *IEEE Internet Computing* 14.5 (Oct. 2010), pp. 60–69 (cit. on p. 27).

[Rob94] Warren Robinett. „Interactivity and Individual Viewpoint in Shared Virtual Worlds: The Big Screen vs. Networked Personal Displays“. In: *SIGGRAPH Comput. Graph.* 28.2 (May 1994), pp. 127–130 (cit. on p. 6).

[Rom+10] Salvador J. Romero, Luis Fernandez-Luque, José L. Sevillano, and Lars Vognild. „Open Source Virtual Worlds and Low Cost Sensors for Physical Rehab of Patients with Chronic Diseases“. In: *Electronic Healthcare*. Ed. by Patty Kostkova. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 27. Springer Berlin Heidelberg, Jan. 2010, pp. 76–79 (cit. on p. 22).

[Ros95] Sheldon M. Ross. *Stochastic Processes*. 2nd ed. Wiley, Jan. 1995 (cit. on p. 58).

[RP09] Malcolm Ryan and Yusuf Pisan. *IE ’09: Proceedings of the Sixth Australasian Conference on Interactive Entertainment*. New York, NY, USA: ACM, 2009 (cit. on p. 4).

[RPW09] Muhammad Azizur Rahman, Algirdas Pakštas, and Frank Zhigang Wang. „Network modelling and simulation tools“. In: *Simulation Modelling Practice and Theory* 17.6 (July 2009), pp. 1011–1031 (cit. on p. 85).

[RSR08] M. Ries, P. Svoboda, and M. Rupp. „Empirical study of subjective quality for Massive Multiplayer Games“. In: *15th International Conference on Systems, Signals and Image Processing, 2008. IWSSIP 2008*. June 2008, pp. 181–184 (cit. on p. 7).

[s15] Group ‘s’. *Cyber1*. `http://www.cyber1.org/`. (visited on 2015-10-24). 2015 (cit. on p. 5).

[Sal+12] J. Saldana, L. Sequeira, J. Fernandez-Navajas, and J. Ruiz-Mas. „Traffic optimization for TCP-based Massive Multiplayer Online Games“. In: *2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. July 2012, pp. 1–8 (cit. on p. 8).

[Sco79] David W. Scott. „On Optimal and Data-Based Histograms“. In: *Biometrika* 66.3 (Dec. 1979), pp. 605–610 (cit. on p. 51).

[SE79] William Sorlie and Diane L. Essex. „The University of Illinois Basic Medical Sciences PLATO IV Project–An Evaluation“. In: *Journal of Computer-Based Instruction* 5.3 (Feb. 1979), pp. 50–56 (cit. on p. 5).

[SGI93] SGI. *Dogfight Manual Page from SGI IRIX v6.5*. *Link to manual page*. Unix Manual Page. (visited on 2015-10-27). 1993 (cit. on p. 6).

[Sha+14] Shayan Shahand, Mohammad Mahdi Jaghoori, Ammar Benabdelkader, et al. „Computational Neuroscience Gateway: A Science Gateway Based on the WS-PGRADE/gUSE“. In: *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*. Springer, Oct. 2014, pp. 139–150 (cit. on pp. 14, 35).

[She+03] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. „The effect of latency on user performance in Warcraft III“. In: *Proceedings of the 2nd workshop on Network and system support for games*. NetGames ’03. New York, NY, USA: ACM, 2003, pp. 3–14 (cit. on pp. 26, 33).

[SJT06] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen. „VON: a scalable peer-to-peer network for virtual environments“. In: *IEEE Network* 20.4 (July 2006), pp. 22–31 (cit. on p. 21).

[SKR07] P. Svoboda, W. Karner, and M. Rupp. „Traffic Analysis and Modeling for World of Warcraft“. In: *IEEE International Conference on Communications, 2007. ICC ’07*. June 2007, pp. 1612–1617 (cit. on p. 7).

[SM13] Luís Miguel Sequeira and Leonel Caseiro Morgado. „Virtual Archaeology in Second Life and OpenSimulator“. In: *Journal For Virtual Worlds Research* 6.1 (Apr. 2013) (cit. on p. 10).

[Sof93] Id Software. *Doom Press Release*. *Link to John Romero’s blog*. (visited on 2015-11-02). Jan. 1993 (cit. on p. 7).

[Sta12] StatCounter. *Top 5 Operating Systems on Oct 2012 | StatCounter Global Stats*. *Link to site*. (visited on 2012-11-25). 2012 (cit. on p. 28).

[Ste74] M. A. Stephens. „EDF Statistics for Goodness of Fit and Some Comparisons“. In: *Journal of the American Statistical Association* 69.347 (Sept. 1974), pp. 730–737 (cit. on p. 40).

[SW09] Galen R. Shorack and Jon A. Wellner. *Empirical Processes with Applications to Statistics*. SIAM, Sept. 2009 (cit. on p. 39).

[SZ99] Sandeep Singhal and Michael Zyda. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999 (cit. on pp. 1, 9, 18).

[Teaa] HotKeyNet Team. *HotkeyNet Site*. `http://hotkeynet.com/`. (visited on 2012-12-04) (cit. on pp. 32, 110).

[Teab] jME Team. *jMonkeyEngine*. `http://jmonkeyengine.org/`. (visited on 2018-07-09) (cit. on p. 24).

[Tea19a] The Wireshark Team. *tshark - The Wireshark Network Analyzer 2.6.6*. `https://www.wireshark.org/docs/man-pages/tshark.html`. (visited on 2019-02-07). 2019 (cit. on pp. 57, 107).

[Tea19b] The Wireshark Team. *Wireshark · Go Deep*. `https://www.wireshark.org/`. (visited on 2019-02-07). 2019 (cit. on pp. 33, 56, 57, 107).

[Tec] Technopedia. *What is a Sandbox (in Gaming)? - Definition from Technopedia*. https://www.techopedia.com/definition/3952/sandbox-gaming. (visited on 2019-06-23) (cit. on p. 11).

[TL14] Alireza Tavakkoli and D Loffredo. „Lessons from game studies to enhance gamification in education“. In: *WMSCI 2014 - 18th World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings* 2 (Jan. 2014), pp. 44–49 (cit. on p. 10).

[Tot] Viktor T. Toth. *MUD British Legends*. `http://www.british-legends.com/CMS/index.php/play-the-game/other-ways-to-play`. (visited on 2015-10-27) (cit. on p. 5).

[Tre03] Helmuth Trefftz. *Networked Virtual Environments*. *Link to the presentation slides*. Tutorial. (visited on 2015-08-01). Honolulu, Hawaii, Aug. 2003 (cit. on p. 18).

[Tuc59] Howard G. Tucker. „A Generalization of the Glivenko-Cantelli Theorem“. In: *The Annals of Mathematical Statistics* 30.3 (1959), pp. 828–830 (cit. on p. 40).

[Vaa00] A. W. van der Vaart. *Asymptotic statistics*. Cambridge: Cambridge University Press, 2000 (cit. on p. 40).

[VDK13] Mathieu Valero, Raluca Diaconu, and Joaquin Keller. „Manycraft: Massively distributed minecraft“. In: *2013 12th Annual Workshop on Network and Systems Support for Games (NetGames)*. Denver, CO, USA: IEEE, Dec. 2013, pp. 1–3 (cit. on p. 12).

[Vil+10] A. Vilela, M. Cardoso, D. Martins, et al. „Privacy Challenges and Methods for Virtual Classrooms in Second Life Grid and OpenSimulator“. In: *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*. Mar. 2010, pp. 167–174 (cit. on p. 10).

[Vot04] O. Voth. „Gaming technology helps troops learn language“. In: *IEEE Intelligent Systems* 19.5 (Sept. 2004), pp. 4–6 (cit. on p. 3).

[Wat+06] A. F. Wattimena, Robert E. Kooij, J. M. Van Vugt, and O. K. Ahmed. „Predicting the perceived quality of a first person shooter: the Quake IV G-model“. In: *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 42 (cit. on p. 8).

[Wav01] J. M. P. van Waveren. „The quake III arena bot“. In: *Master of Science thesis Delft University of Technology* (2001) (cit. on p. 31).

[WCJ14] Jih-Wei Wu, Ding-Wei Chou, and Jehn-Ruey Jiang. „The Virtual Environment of Things (VEoT): A Framework for Integrating Smart Things into Networked Virtual Environments“. In: *2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. Taipei, Taiwan: IEEE, Sept. 2014, pp. 456–459 (cit. on p. 20).

[Wei+17] Derek Weitzel, Brian Bockelman, Dave Dykstra, Jakob Blomer, and Ren Meusel. „Accessing data federations with CVMFS“. In: *J.Phys.Conf.Ser.* 898.6 (Nov. 2017), p. 062044 (cit. on p. 105).

[WI09] Christian Wagner and Rachael K. F. Ip. „Action Learning with Second Life–A Pilot Study“. In: *Journal of Information Systems Education* 20.2 (2009), pp. 249–258 (cit. on p. 11).

[Wol02] Mark J. P. Wolf. „The Medium of the Video Game“. In: *The Medium of the Video Game*. 1 edition. Austin: University of Texas Press, Feb. 2002, pp. 113–134 (cit. on p. 4).

[XS16] Mingze Xi and Shamus P. Smith. „Supporting path switching for non-player characters in a virtual environment“. In: *2016 IEEE Virtual Reality (VR)*. Greenville, SC, USA: IEEE, Mar. 2016, pp. 315–316 (cit. on p. 21).

[Yan+04] Nicole Yankelovich, William Walker, Patricia Roberts, et al. „Meeting central: making distributed meetings more effective“. In: ACM, Nov. 2004, pp. 419–428 (cit. on p. 21).

[Yan12] Yang Yang. *Understanding Switch Latency, Cisco Nexus 3000 Series Switches*. *Link to paper*. (visited on 2013-04-02). 2012 (cit. on p. 28).

[YJG03] Andy B. Yoo, Morris A. Jette, and Mark Grondona. „SLURM: Simple Linux Utility for Resource Management“. In: *Job Scheduling Strategies for Parallel Processing*. Ed. by Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 44–60 (cit. on p. 104).

[YK13] Amir Yahyavi and Bettina Kemme. „Peer-to-peer Architectures for Massively Multiplayer Online Games: A Survey“. In: *ACM Comput. Surv.* 46.1 (July 2013), 9:1–9:51 (cit. on pp. 9, 18).

[ZA04] Sebastian Zander and Grenville Armitage. „Empirically measuring the QoS sensitivity of interactive online game players“. In: *Proc. ATNAC*. 2004, pp. 511–518 (cit. on p. 8).

[ZA05] S. Zander and G. Armitage. „A traffic model for the Xbox game Halo 2“. In: *Proceedings of the international workshop on Network and operating systems support for digital audio and video*. 2005, pp. 13–18 (cit. on p. 36).

List of Figures

List of Tables

Acronyms

ACF autocorrelation function.

AHK AutoHotKey.

AI artificial intelligence.

AOL America Online.

ARMA autoregressive-moving-average.

BYOC bring your own computer.

CDF cumulative distribution function.

CLI command-line interface.

CSMA carrier sense multiple access.

CSV comma-separated value.

CVMFS CernVM File System.

DDoS distributed denial-of-service.

DIS Distributed Interactive Simulation.

DIVE Distributed Interactive Virtual Environment.

DoD Department of Defense.

ECDF	empirical cumulative distribution function.
FCS	frame check sequence.
FPS	first-person shooter.
GoF	goodness of fit.
GPU	graphics processing unit.
GUI	graphical user interface.
HKN	HotKeyNet.
HPC	high-performance computing.
HTC	high-throughput computing.
IAT	inter-arrival time.
IDT	inter-departure time.
IoT	Internet of things.
JDK	Java Development Kit.
JRE	Java Runtime Environment.
LAN	local area network.
LSVE	large-scale virtual environment.
MLE	maximum likelihood estimation.
MMORPG	massively multiplayer online role-playing game.
MUD	Multi-User Dungeon.

MWTB	Mounted Warfare TestBed.
NIC	network interface controller.
NVE	networked virtual environment.
OO	object oriented.
OSIAT	object synchronisation inter-arrival time.
OSPS	object synchronisation packet size.
OWL	Open Wonderland.
OWTB1	Open Wonderland TestBed 1.
OWTB2	Open Wonderland TestBed 2.
P2P	peer-to-peer.
PCC	Pearson correlation coefficient.
PDF	probability density function.
PLATO	Programmed Logic for Automatic Teaching Operations.
PS	packet size.
QoE	quality of experience.
QoS	quality of service.
RPG	role-playing game.
RTS	real-time strategy.
SGI	Silicon Graphics, Inc..

SIF	Singularity Image Format.
SP1	Service Pack 1.
VAE	Virtual Army Experience.
VBR	variable bitrate.
VBS1	Virtual Battlefield Systems 1.
VR	virtual reality.
WoW	World of Warcraft.

Colophon

The research presented in this thesis was performed at the Department of Computer Technology and Architecture, part of the Escuela Superior de Ingeniería Informática, University of Seville, in collaboration with University of Pisa, Italy.

Financial support for travel, conference attendance, publications and other logistics were provided by Consejería de Educación of Junta de Andalucía through the program *Incentivos para la formación de personal docente e investigador predoctoral en áreas de conocimiento deficitarias* and the Department of Computer Technology and Architecture, University of Seville.

Special thanks to the *Cátedra Telefónica* for its support during the realisation of the present work.

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The bibliography has been managed using *Zotero*.

Declaration

I Juan Luis Font Calvo, student of the Department of Computer Technology and Architecture of the University of Seville, declare that I have developed and written the enclosed PhD Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The PhD Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Sevilla, July 2019

Juan Luis Font Calvo