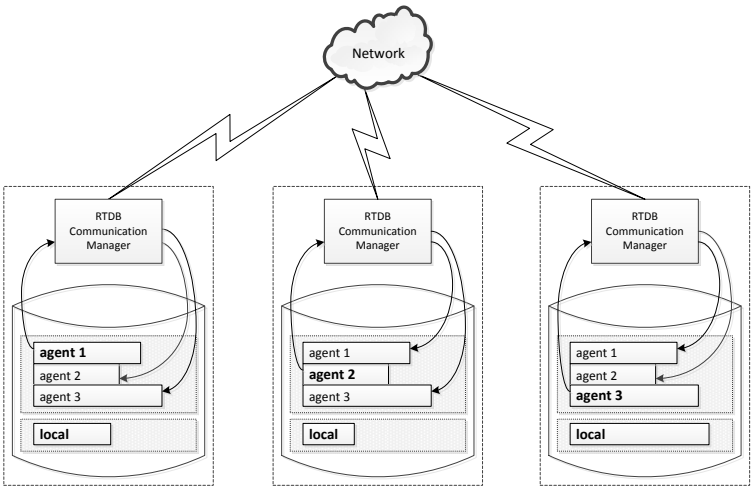
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2014

Frederico Miguel do Céu Marques dos Santos

Arquitectura para coordenação em tempo-real de múltiplas unidades móveis autónomas

Architecture for real-time coordination of multiple autonomous mobile units



Universidade de Aveiro
2014

Departamento de Electrónica, Telecomunicações e Informática

Frederico Miguel do Céu Marques dos Santos

Arquitectura para coordenação em tempo-real de múltiplas unidades móveis autónomas

Architecture for real-time coordination of multiple autonomous mobile units

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Electrotécnica, realizada sob a orientação científica do Doutor Luís Miguel Pinho de Almeida, Professor Associado da Faculdade de Engenharia da Universidade do Porto e co-orientação do Doutor Luís Filipe de Seabra Lopes, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Apoio financeiro da FCT no âmbito do III Quadro Comunitário de Apoio, através da bolsa de Doutoramento SFRH/BD/29839/2006

Apoio financeiro através do 7º Programa Quadro da Comissão Europeia através do projecto ArtistDesign ICT-NoE-214373

dedicatória

Catarina,
Joana e Bruno

o júri / the jury

presidente / president

Doutor Amadeu Mortágua Velho da Maia Soares
Professor Catedrático da Universidade de Aveiro
(por delegação do Reitor da Universidade de Aveiro)

vogais / examiners commitee

Doutora Margarita Marcos-Muñoz
Professora Catedrática da Universidade do País Basco – Espanha

Doutor Daniel Mossé
Professor Catedrático da Universidade de Pittsburgh – E.U.A.

Doutor Luís Miguel Pinho de Almeida
Professor Associado da Faculdade de Engenharia da Universidade do Porto
(orientador)

Doutor António Paulo Gomes Mendes Moreira
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Doutor Luís Filipe de Seabra Lopes
Professor Associado da Universidade de Aveiro
(co-orientador)

Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Doutor Artur José Carneiro Pereira
Professor Auxiliar da Universidade de Aveiro

agradecimentos

Durante esta longa jornada, muitas foram as pessoas que intervieram, de alguma forma, no desenvolvimento do trabalho. A todos, sem excepção, fica o meu grande agradecimento, pois sem a vossa ajuda nada disto teria sido possível.
No entanto, dada a sua relevância para o presente trabalho, gostaria de individualizar os meus agradecimentos a:

Luís Almeida, que me orientou, motivou, ajudou e acompanhou em todas as fases deste trabalho. A ele fica o meu muito reconhecido agradecimento: Obrigado Luís!

Luís Seabra Lopes, meu co-orientador e primeiro *team-leader* da equipa CAMBADA de futebol robótico da Universidade de Aveiro, pela ajuda prestada.

Daniel Mossé, que me recebeu durante três semanas na Universidade de Pittsburgh, EUA, assim como a Ihsan Qazi pelas discussões técnicas.

Gustavo Corrente, Ricardo Sequeira e João Alex Cunha, os Cambadistas *student*, companheiros de longas jornadas e muitas aventuras.

José Luís Azevedo e Bernardo Cunha, os Cambadistas *senior* que sempre acompanharam a CAMBADA.

Nuno Figueiredo, Nelson Filipe e Daniel Martins, o paternalista, o vegetariano e o vassouras.

Toda a restante equipa CAMBADA, que proporcionou ao longo destes anos, grandes momentos e muitas vitórias. Viva a CAMBADA!

Paulo Pedreiras, Valter Silva, Ricardo Marau e Rui Santos, os puros e duros da 319.

Artur Pereira, Tiago Nunes, Milton Gregório e David Ferreira, da Universidade de Aveiro, pela colaboração no desenvolvimento de um *parser* para criação automática dos ficheiros de configuração da RTDB.

Luís Oliveira, Ana Pereira e Fábio Silva, da Faculdade de Engenharia da Universidade do Porto, pela ajuda nos testes de sincronização de relógio.

Aos meus colegas do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia de Coimbra.

Catarina Santos, minha esposa, por toda a paciência.

A todos um bem-haja pelo vosso apoio

palavras-chave

sistemas multi-robô, sistemas de tempo-real, partilha de informação, comunicação sem fios, protocolos TDMA, bases de dados distribuídas

resumo

O interesse na utilização de equipas multi-robô tem vindo a crescer, devido ao seu potencial para cooperarem na resolução de vários problemas, tais como salvamento, desminagem, vigilância e até futebol robótico. Estas aplicações requerem uma infraestrutura de comunicação sem fios, em tempo real, suportando a fusão eficiente e atempada dos dados sensoriais de diferentes robôs bem como o desenvolvimento de comportamentos coordenados. A coordenação de vários robôs autónomos com vista a um dado objectivo é actualmente um tópico que suscita grande interesse, e que pode ser encontrado em muitos domínios de aplicação. Apesar das diferenças entre domínios de aplicação, o problema técnico de construir uma infraestrutura para suportar a integração da percepção distribuída e das acções coordenadas é similar. O problema torna-se mais difícil à medida que o dinamismo dos robôs se acentua, por exemplo, no caso de se moverem mais rápido, ou de interagirem com objectos que se movimentam rapidamente, dando origem a restrições de tempo-real mais apertadas.

Este trabalho centrou-se no desenvolvimento de arquitecturas computacionais e protocolos de comunicação sem fios para suporte à partilha de informação e à realização de acções coordenadas, levando em consideração as restrições de tempo-real. A tese apresenta duas afirmações principais. Em primeiro lugar, apesar do uso de um protocolo de comunicação sem fios que inclui mecanismos de arbitragem, a auto-organização das comunicações reduz as colisões na equipa, independentemente da sua composição em cada momento. Esta afirmação é validada em termos de perda de pacotes e latência da comunicação. Mostra-se também como a auto-organização das comunicações pode ser atingida através da utilização de um protocolo TDMA reconfigurável e adaptável sem sincronização de relógio.

A segunda afirmação propõe a utilização de um sistema de memória partilhada, com replicação nos diferentes robôs, para suportar o desenvolvimento de mecanismos de percepção distribuída, fusão sensorial, cooperação e coordenação numa equipa de robôs. O sistema concreto que foi desenvolvido é designado como Base de Dados de Tempo Real (RTDB). Os dados remotos, que são actualizados de forma transparente pelo sistema de comunicações auto-organizado, são estendidos com a respectiva idade e são disponibilizados localmente a cada robô através de primitivas de acesso eficientes. A RTDB facilita a utilização parcimoniosa da rede e bem como a manutenção de informação temporal rigorosa. A simplicidade da integração da RTDB para diferentes aplicações permitiu a sua efectiva utilização em diferentes projectos, nomeadamente no âmbito do RoboCup.

keywords

multi-robot systems, real-time systems, information sharing, wireless communication, TDMA protocols, distributed databases

abstract

Interest on using teams of mobile robots has been growing, due to their potential to cooperate for diverse purposes, such as rescue, de-mining, surveillance or even games such as robotic soccer. These applications require a real-time middleware and wireless communication protocol that can support an efficient and timely fusion of the perception data from different robots as well as the development of coordinated behaviours. Coordinating several autonomous robots towards achieving a common goal is currently a topic of high interest, which can be found in many application domains. Despite these different application domains, the technical problem of building an infrastructure to support the integration of the distributed perception and subsequent coordinated action is similar. This problem becomes tougher with stronger system dynamics, e.g., when the robots move faster or interact with fast objects, leading to tighter real-time constraints.

This thesis work addressed computing architectures and wireless communication protocols to support efficient information sharing and coordination strategies taking into account the real-time nature of robot activities. The thesis makes two main claims. Firstly, we claim that despite the use of a wireless communication protocol that includes arbitration mechanisms, the self-organization of the team communications in a dynamic round that also accounts for variable team membership, effectively reduces collisions within the team, independently of its current composition, significantly improving the quality of the communications. We will validate this claim in terms of packet losses and communication latency. We show how such self-organization of the communications can be achieved in an efficient way with the Reconfigurable and Adaptive TDMA protocol.

Secondly, we claim that the development of distributed perception, cooperation and coordinated action for teams of mobile robots can be simplified by using a shared memory middleware that replicates in each cooperating robot all necessary remote data, the Real-Time Database (RTDB) middleware. These remote data copies, which are updated in the background by the self-organizing communications protocol, are extended with age information automatically computed by the middleware and are locally accessible through fast primitives. We validate our claim showing a parsimonious use of the communication medium, improved timing information with respect to the shared data and the simplicity of use and effectiveness of the proposed middleware shown in several use cases, reinforced with a reasonable impact in the Middle Size League of RoboCup.

Contents

List of Figures

List of Tables

Symbols

$\boldsymbol{\beta}$	Increment in period during the scan mode (*initial version of Adaptive TDMA*) (time)
$\boldsymbol{\Delta}$	Validity window for transmission delays inside each slot (time)
$\boldsymbol{\epsilon}$	Relative validity window (fraction of slot width)
$\boldsymbol{K}$	Current number of nodes in the team
$\boldsymbol{M}$	Current number of slots per round
$\boldsymbol{N}$	Maximum number of nodes in the team
$\boldsymbol{n_{dup}}$	Update period of a given RTDB item (number of rounds)
$\boldsymbol{\Omega}$	Target maximum network utilization (fraction)
$\boldsymbol{R}$	Maximum number of rounds to synchronize using scan mode (*initial version of Adaptive TDMA*)
$\boldsymbol{T_{join}}$	Joining latency for arriving team members (time)
$\boldsymbol{T_{rcpp}}$	Producer period of a given RTDB item (time)
$\boldsymbol{T_{reconf}}$	Total reconfiguration latency caused by a joining node (time)
$\boldsymbol{T_{tup}}$	Team update period - target TDMA round (time)
$\boldsymbol{\tilde{T}_{tup}}$	Effective team update period (time)
$\boldsymbol{T_{wt}}$	Transmission time of an RTDB packet
$\boldsymbol{T_{xwin}}$	Width of the TDMA slot (time)

Acronyms

AMP	Alternate MAC/PHY
AP	Access Point
API	Application Programming Interface
BSD	Berkeley Software Distribution
BSS	Basic Service Set
CAMBADA	Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture
CORBA	Common Object Request Broker Architecture
CORBA/e	CORBA for *embedded*
COTS	Commercial Off-The-Shelf
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSS	Chirp Spread Spectrum
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
DDS	Data Distribution Service
DDSI	DDS Interoperability
DFS	Dynamic Frequency Selection
DIFS	Distributed Coordination Function Inter Frame Spacing
DSSS	Direct-Sequence Spread Spectrum
DTIM	Delivery Traffic Information Map
EDCA	Enhanced Distributed Channel Access
EDR	Enhanced Data Rate
FFD	Full-Function Device
FHSS	Frequency Hopping Spread Spectrum
GPL	General Public License

HCCA	Hybrid Coordination Function Controlled Channel Access
HS/DSSS	High Speed Direct-Sequence Spread Spectrum
I/O	Input/output
ICE	Internet Communications Engine
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Media Access Control layer
MIRO	Middleware for Robots
MSL	Middle Size League
MTU	Maximum Transmission Unit
NAV	Network Allocation Vector
OCERA	Open Components for Embedded Real-Time Applications
OFDM	Orthogonal Frequency-Division Multiplexing
OMG	Object Management Group
ORB	Object Request Broker
ORCA	Open Robot Controller Architecture
ORTE	OCERA Real-Time Ethernet
OSI	Open Systems Interconnection
PAN	Personal Area Network
PC	Publisher-Consumer
PHY	Physical layer
PI	Proportional-Integral
PLCP	Physical Layer Convergence Protocol
PS	Publish-Subscriber
PSK	Phase-Shift Keying
QoS	Quality of Service
RF	Radio Frequency
RFD	Reduced-Function Device
ROS	Robot Operating System
RPC	Remote Procedure Call

RT	Real-Time
RT-Component	Real-Time Component
RT-CORBA	Real Time CORBA
RT-Middleware	Real-Time Middleware
RTDB	Real-Time Database
RTPS	Real-Time Publish Subscribe
RTS	Request To Send
SIFS	Short Inter Frame Spacing
SIG	Special Interest Group
SIR	Signal-to-Interference Ratio
SLAM	Simultaneous Localization And Mapping
SOAP	Simple Object Access Protocol
SPL	Standard Platform League
SSL	Soccer Simulation League
TAO	The ACE ORB
TDMA	Time Division Multiple Access
TIM	Traffic Indication Map
U-NII	Unlicensed National Information Infrastructure
UDP	User Datagram Protocol
UML	Unified Modeling Language
USA	United States of America
UWB	Ultra-Wide Band
V-FTT	Vehicular Flexible Time-Triggered
W3C	World Wide Web Consortium
WAVE	Wireless Access in Vehicular Environments
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
XML	Extensible Markup Language

Chapter 1

Introduction

Coordinating several autonomous mobile robotic agents in order to achieve a common goal continues to be an active topic of research [55, 93]. The use of several cooperative robots can increase the system effectiveness, with respect to a single autonomous robot or to a team of non cooperating robots.

Instead of using a single powerful robot, a multi-robot solution can be easier and cheaper, can provide flexibility to task execution and can make the system tolerant to possible robot failures. A multi-robot system can better perform a mission in terms of time and quality, can achieve tasks not executable by a single robot and can take advantages of distributed sensing and actuation.

Examples of applications are:

- Search and rescue in catastrophic situations, where the utilization of teams of robots allows to explore a large area and/or areas inaccessible to persons;
- Transportation of large/indivisible objects, when more than one robot is necessary to transport such objects and the coordination of the movements is a critical requirement;
- Surveillance, where multiple robots can cover a wider area, either in land, water or sky.

The technical problem of building an infrastructure to support the perception integration for a team of robots and subsequent coordinated action is common to the above applications, where robots must interact among them using communication to exchange information and take decisions based on the team perception of the environment, e.g., fusing sensory data from all the robots in the team.

1.1 Multi-Robot systems

Multi-robot systems are becoming one of the most important areas of research in robotics, due to the challenging nature of the involved research and to the multiple potential applications to areas such as autonomous and mobile sensor networks, building surveillance, transportation of large objects, air and underwater pollution monitoring, forest fire detection, transportation systems, or search and rescue in large-scale disasters. Even problems that can be handled by a single multi-skilled robot may benefit from the alternative usage of a robotic team, since robustness and reliability can often be increased by combining several robots which are individually less robust and reliable or have distinct sensors and actuators.

It is possible to find similar examples in human work: several people in line are able to move a bucket, from a water source to a fire, faster and with less individual effort. Also, if one or more of the individuals leaves the team, the task can still be accomplished by the remaining ones, even if slower than before. Another example is the surveillance of a large area by several people. If adequately coordinated, the team is able to perform the job faster and possible with reduced cost than a single person carrying out all the work, especially if the cost of moving over large distances is prohibitive.

The use of multi-robot systems to reach a common goal involves cooperative actions, i.e., robots acting as a team. In turn, achieving cooperative behavior must rely on an infrastructure that encompasses such concepts as robot heterogeneity/homogeneity, the ability of a given robot to recognize and model other robots, and communication ability.

1.1.1 Cooperation

The Merriam-Webster on-line dictionary[1] defines *cooperate* as:

"to act or work with another or others"
"to associate with another or others for mutual benefit"

The meaning of cooperation among robots was defined in [15] as:

- *A joint collaborative behavior that is directed toward some goal in which there is a common interest or reward*;
- *A form of interaction, usually based on communication*;

[1] `http://www.merriam-webster.com`

- *Joining together for doing something that creates a progressive result such as increasing performance or saving time.*

We propose our own definition for cooperative behavior as follows: "*Given some task specified by a designer, a multiple-robot system displays cooperative behavior if, due to some underlying mechanism (i.e., the mechanism of cooperation), there is an increase in the total utility of the system*".

This definition makes it clear that cooperation does not only imply interaction but a *constructive interaction* in the sense that it leads to consistent performance improvements. With cooperation, multiple robots can exchange their roles, dynamically, according to physical positioning or ambient constraints, or they can adapt their positions according to each robot specific sensor and/or actuator systems.

In Robotics, cooperation can take place in different areas [4]:

- **Cooperative perception** – The robotic system, in a distributed way, is able of retrieve and interpret sensor data, leading to individual and/or collective understanding of the environment. A significant number of tasks deeply rely on cooperative perception, e.g., cooperative situation assessment, cooperative tracking or cooperative learning from sensor information;

- **Cooperative learning** – The ability for a team of robots to learn the features of the environment and, when relevant, opponent models. Moreover, a team of agents should be able to learn collective behaviors, such as strategies to pursue their goals in the environment, in the face of competitors. Learning in multi-robot systems is affected by specific challenges like multiple goals, noisy perception and actions, and inconsistencies between the internal states and environment models of the individual robots;

- **Cooperative planning and execution** – A feature of multi-robot domains is the uncertainty arising from both perception and action. Another uncertainty is brought by the presence of other agents that can interfere, while pursuing their own goals or even by having competing goals. In this context, the robotic team must deal with external factors and must be able to plan tasks and decisions, in a dynamic and distributed/decentralized way.

Moreover, cooperation within the team must continue, even with the failure of one or more robots. Also, a robot that leaves the team, e.g. by moving too far to be in the range of the others, must possess autonomous capabilities, allowing to continue operating alone.

1.1.2 Infrastructure to support cooperation: the middleware

Despite advances in recent years, multi-robot systems remain complex, with the control and coordination of these systems being a challenging task. These complexities make the development of components for multi-robot applications non-trivial and failure prone.

The difficulties in developing multi-robot systems arise from the following:

- Rapid changes in sensors, actuators and computer technologies lead to increased pressure towards new robot capabilities, e.g. new sensor and actuator systems may require more sophisticated signal processing algorithms;
- Robotic systems are inherently distributed, i.e. sensors and actuators are distributed over interconnected subsystems, and in multi-robot systems the distribution scales up;
- Robots, components and processes require to interact in an efficient way;
- Each individual robot requires a tight coupling of its sensing, processing, acting and abilities;
- Each robot must also be autonomous and incorporate a myriad of algorithms such as Simultaneous Localization And Mapping (SLAM), obstacle avoidance, navigation primitives and vision processing algorithms, to mention a few;
- The robotic team must continue operation in a cooperative way in spite of individual robot crashes and asynchronous restarts as well as robots that move out of reach and rejoin later.

Consequently, the development of robots capable of sophisticated decision-making, both individually and as a team, is rather complex. The time and work required to come up with such a robot system is high. A robot programmer needs to be well informed in a number of engineering fields, such as signal processing, control, electricity and electronics, in addition to standard computer science and artificial intelligence background. This diversity typically requires the collaboration of different developers and groups with specific expertise in those individual areas, further exacerbating the complexity of building robotic systems.

Concerning the specific case of the software for robotic systems, there is currently a trend towards developing it on top of a middleware layer. This is connectivity software that consists of a set of enabling services that allow multiple modular processes (modules/components) running on board of a robot, and possibly some of them off board, to interact across a network [16]. This connectivity software also supports component reuse, particularly the

reuse of modules for common problems in Robotics, e.g. Extended Kalman filters and SLAM algorithms. This 'plug and play' approach has a great potential for the future development of robotic applications but requires the definition of adequate standards.

In general, integrating modules and components possibly developed separately and with different technologies is not an easy task. A standard middleware integrates this diversity by providing abstract interfaces and transparent communication protocols that are platform independent as much as possible. Then, programmers can concentrate on building the modules/components independently of the rest of the system, as well as on their integration and development of the higher level applications. This will lead to faster development of teams of cooperative robots as well as to increased robot intelligence and to multi-robot systems that are capable of performing more complex tasks.

In this work we will focus on simplifying the development of cooperative behaviors while making such cooperation more efficient and effective with an adequate use of communications.

1.2 The thesis

This thesis makes two main claims. Firstly, despite the use of a wireless communication protocol that includes arbitration mechanisms, namely the standard defined by Institute of Electrical and Electronics Engineers (IEEE) as IEEE 802.11, the self-organization of the team communications in a dynamic round that also accounts for variable team membership effectively reduces collisions within the team, independently of its current composition, significantly improving the quality of the communications. We will validate this claim in the scope of infrastructured communications, i.e., through an Access Point (AP), with respect to the number of packet losses and communication latency. We show how such self-organization of the communications can be achieved in an efficient way with the Reconfigurable and Adaptive Time Division Multiple Access (TDMA) protocol, without making use of additional control mechanisms, e.g., for explicit transmission control or even clock synchronization, and being resilient to external uncontrolled traffic as that generated by nodes outside the team.

Secondly, we claim that the development of distributed perception, cooperation and coordinated action for teams of mobile robots can be simplified by using a shared memory middleware that replicates in each cooperating robot all necessary remote data, the Real-Time Database (RTDB). These remote data images are locally accessed with fast primitives, decoupled from the communications, they are extended with age information automatically computed by the middleware, and they are updated in the background by the self-organizing communications protocol referred above, at a convenient refresh rate to maintain data validity. We validate our claim showing a parsimonious use of the communication medium,

an improved timing information with respect to the shared data and the simplicity of use and effectiveness of the proposed middleware shown in several use cases, reinforced with a reasonable impact in the Middle Size League (MSL) of RoboCup.

1.3 Contributions

The main contributions resulting from this work were referred above in the thesis, namely the wireless communication protocol for infrastructured IEEE 802.11 networks called Reconfigurable and Adaptive TDMA[2] and the RTDB lightweight middleware.

The Reconfigurable and Adaptive TDMA wireless communication protocol is responsible for synchronizing the transmissions of the robotic team members in a cycle without resorting to a clock synchronization service, reducing intra-team collisions.

This protocol can be directly applied on any infrastructured wireless network and requires no prior configuration of the robots in the team, which automatically (re)synchronize as soon as they are launched or restarted.

The Reconfigurable and Adaptive TDMA protocol has been presented in the following publications:

- Frederico Santos, Luís Almeida and Luís Seabra Lopes. Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. In *Proceedings of ETFA 2008 - 13th IEEE Conference on Emerging Technologies and Factory Automation*, p. 1197-1204, Hamburg, Germany, September 2008. [Google Scholar citations: 27]

- Frederico Santos, Gustavo Corrente, Luís Almeida, Nuno Lau and Luís Seabra Lopes. Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. In *Proceedings of EPIA 2007 - 13th Portuguese Conference on Artificial Intelligence*, p. 657-665, Guimarães, Portugal, December 2007.[3]

[2] In previous publications of this protocol we have referred to it as *RA-TDMA*. However, in this dissertation we preferred keeping the full name since that acronym has been previously used for *Random-Access TDMA*, which is an unrelated technique.

[3] This and the previous references ended up with the same title by mistake, despite being two substantially different papers. The previous one has an in depth description and assessment of the protocol focusing on the self-reconfiguration feature while this one is a short paper addressing both the RTDB and the communication protocol, and the importance of the latter to the behaviors developed over the former.

- Frederico Santos and Luís Almeida. On the effectiveness of IEEE802.11 broadcasts for soft real-time communication. In *Proceedings of RTN'05 - 4th International Workshop on Real-Time Networks*, Palma de Mallorca, Spain, July 2005. [Google Scholar citations: 3]

- Frederico Santos, Luís Almeida, Tullio Facchinetti, Paulo Pedreiras, and Luís Seabra Lopes, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents", In *Proceedings of WACERTS'04 - International Workshop on Architectures for Cooperative Embedded Real-Time Systems, in conjunction with the RTSS04 - 25th IEEE International Real-Time Systems Symposium*, Lisbon, Portugal, December 2004. [Google Scholar citations: 28]

It is worth noting that the publication at ETFA 2008 (the 13th IEEE Conference on Emerging Technologies and Factory Automation) received the conference Best Paper Award.

The second most relevant contribution is the RTDB middleware which is based on a shared memory model that is replicated in all nodes and it is rather lightweight when compared with other standard distributed computing middlewares such as Common Object Request Broker Architecture (CORBA) or Data Distribution Service (DDS). Moreover, it also implies a parsimonious use of the communciation medium when compared with other typical centralized implementations of shared memory such as the Blackboard, which need two network transactions to accomplish any effective data transfer, while the RTDB does so with a single network transaction per node.

From the point of view of the nodes, the RTDB behaves as a proxy of the remote data, providing immediate local access. This allows removing the communication latencies from the execution time of the programs that use the remote data, contributing to improve their temporal behavior.

On the other hand, the RTDB also presents some limitations that arise from the options taken in its conception. For example, it is made for systems in which all the machines have a similar hardware architecture and run a software in which the data types are similar, too.

The RTDB has been presented in the following publications:

- Frederico Santos, Luís Almeida, Paulo Pedreiras, Luís Seabra Lopes. A real-time distributed software infrastructure for cooperating mobile autonomous robots. In *Proceedings of ICAR 2009 - 14th International Conference on Advanced Robotics*, p. 923-928. Munich, Germany, June 2009. [Google Scholar citations: 6]

- Luís Almeida, Frederico Santos, Tullio Facchinetti, Paulo Pedreiras, Valter Silva, and Luís Seabra Lopes. Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project. In Cevdet Aykanat, Tugrul Dayar and Ibrahim Korpeoglu, editors, *ISCIS - 19th International Symposium on Computer and Information Sciences*, volume 3280 of *Lecture Notes in Computer Science*, pages 876-886. Springer-Verlag, 2004. [Google Scholar citations: 42]

Finally, note that the two main contributions mentioned above were developed in the scope of the Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture (CAMBADA) RoboCup MSL robotic soccer team from the University of Aveiro. In that scope, the work in this thesis also led to another contribution, namely a clear assessment and understanding of the wireless communications within that competition, influencing the rules of the games at a certain point in time, particularly with respect to bandwidth limitations per team.

This work, in the scope of the RoboCup MSL, was presented in the following publications:

- Frederico Santos, Luís Almeida, Luís Seabra Lopes, José L. Azevedo, M. Bernardo Cunha. Communicating among robots in the RoboCup Middle-Size League. In *RoboCup Symposium 2009*, Graz, Austria, July 2009 (Lecture Notes in Computer Science LNCS 5949, p. 320-331, Springer 2010, ISBN 978-3-642-11875-3). [Google Scholar citations: 12]

- António J. R. Neves, José Luís Azevedo, M. Bernardo Cunha, Nuno Lau, João Silva, Frederico Santos, Gustavo Corrente, Daniel A. Martins, Nuno Figueiredo, Artur Pereira, Luís Almeida, Luís Seabra Lopes, Armando J. Pinho, João Rodrigues and Paulo Pedreiras. CAMBADA soccer team: from robot architecture to multiagent coordination. in *Robot Soccer*, Vladan Papić (ed), INTECH, p. 19-45, ISBN 978-953-307-036-0, January 2010. available online: `http://www.intechopen.com/books/robot-soccer`. [Google Scholar citations: 7]

Moreover, the RTDB middleware with the Reconfigurable and Adaptive TDMA protocol is an open source project, freely available at `http://code.google.com/p/rtdb/` that was adopted by other RoboCup MSL teams, such as the Tech United from the Technical University of Eindhoven [1], the NuBot from the National University of Defense Technology in China and SocRob from the Instituto Superior Técnico – Technical University of Lisbon [63]. The SPL Portuguese Team from Universities of Aveiro, Porto and Minho [70] with a team of NAOs competing in the RoboCup Standard Platform League (SPL), and CAMBADA@Home from University of Aveiro [7] with an autonomous robot for helping people with health problems and reduced capabilities, also use the developed middleware.

From outside the RoboCup competitions, the Rota project with the Zinguer autonomous robot [61], from University of Aveiro, competes in the Autonomous Driving League of the Portuguese Robotics Festival[4] [54], use the RTDB middleware for local inter-process communication, only, but extended with the Reconfigurable and Adaptive TDMA for debugging purposes.

Finally, a three years long national research project, *PCMMC - Perception-Driven Coordinated Multi-Robot Motion Control*, developed by the Technical University of Lisbon, University of Porto and Polytechnic Institute of Porto, ended recently and was funded by FCT (PTDC/EEA-CRO/100692/2008). It focused on autonomous team formation control for improved global perception and also used the RTDB middleware with the Reconfigurable and Adaptive TDMA protocol as architectural central hub to share data and support the cooperation[5] [72, 73].

1.4 Structure of the dissertation

This chapter outlined the scope of this dissertation and briefly addressed the need for communication and middleware to provide coordination and cooperation in multi-robot systems. To support our thesis, the dissertation is organized as follows:

Chapter 2 – Presents an overview of some wireless technologies that can be used in multi-robot systems providing a coverage of up to 100 meters. Both IEEE standard and non-standard protocols are presented and discussed, concluding with a comparison of the currently most well known wireless standards for personal- and local-area networks.

Chapter 3 – Discusses the middleware challenges in robotics and presents some of the currently most used middlewares. Primarily developed to connect applications to robot sensor and actuator systems, these middlewares include many other features to facilitate and reduce the time of development.

Chapter 4 – Presents one of the current initiatives that promote scientific research in Robotics, through competition, using real world scenarios, namely RoboCup. One of the competitions of RoboCup that focuses on multi-robot coordination is the MSL where two teams of 5 fully autonomous robots each play soccer against each other. This chapter characterizes the wireless communications in this competition, highlighting

[4] `http://www.spr.ua.pt/fnr/`
[5] `http://welcome.isr.ist.utl.pt/project/index.asp?accao=showproject&id_project=157`

problems and their sources, and proposing solutions. This chapter presents, in fact, the motivation for the work in this thesis, clearly defining the problems and identifying directions that set the grounds for the work that follows.

Chapter 5 – Discusses the Reconfigurable and Adaptive TDMA wireless communication protocol, which is used to share data periodically among the robots in the team. Special attention is devoted to the adaptation of the protocol to external interferences and the automatic reconfiguration due to changes in number of team members.

Chapter 6 – Introduces the RTDB middleware. It addresses its software architecture, implementation, usability, and support to timing information, namely age of the data at the moment of retrieval. The remote data inside the RTDB at each node is refreshed in the background, transparently to the applications, by the protocol defined in the previous chapter.

Chapter 7 – Presents experimental results concerning the Reconfigurable and Adaptive TDMA protocol. An extensive profiling is shown to validate the first claim of the Thesis. In particular, comparative results show the benefits of using a synchronized TDMA approach to the communications within the team, particularly concerning transmission latency and packet losses. Low values in these figures of merit are essential to support the RTDB middleware consistency and consequently an effective integration of distributed sensor data and cooperative behaviors. Moreover, this chapter also shows a thorough comparison with a traditional TDMA implementation based on clock synchronization using the Chrony service [67]. Finally, it shows extensive logs of RoboCup MSL games that show the protocol operation in a real scenario focusing on the tracking of the membership information and associated protocol reconfigurations and packet losses.

Chapter 8 – Describes the CAMBADA robotic soccer team case study, focusing on supporting cooperation, coordination, strategy and debug of team behaviors. The team software is built on top of the RTDB middleware with the Reconfigurable and Adaptive TDMA communication protocol behind. This chapter presents a qualitative validation of the second claim of the Thesis highlighting the simplicity of developing and debugging cooperative behaviors on top of the RTDB. A further indirect validation arises from the results of the CAMBADA team along its history and the middleware adoption by other teams.

Chapter 9 – Presents the conclusions of this dissertation, discussing the validation of the Thesis and its contributions, ending with a few open issues in this research that remain open for future work.

Chapter 2

Wireless communications

The coordination of multi-robot systems requires data exchange using a communication network. The information that is shared by each robot in a team of mobile robots typically includes the perceived properties of the surrounding objects, acquired by sensors mounted on the robot. By integrating the perceptions of multiple robots it is possible to build a more accurate and complete *world state*. This makes possible, for example, that each element makes decisions based on information gathered by the sensors of other robots.

The mobility of robots imposes wireless communications, bringing extra constraints such as their limited range, throughput and availability. The amount of information shared, the number of robots within the team and the team distribution in space, are some of the factors directly influencing the choice of the wireless communication system.

Wireless network systems can be broadly divided in two main groups. One group follows the standards set by the IEEE, or other standardization bodies, which results in well defined systems in terms of Physical layer (PHY) and of Media Access Control layer (MAC), Open Systems Interconnection (OSI) model layer 1 and 2, respectively [104]. The other group includes non-standard communication systems (a) in part, i.e., system derived from a standard with slightly modified characteristics, or (b) in whole, i.e., with complete custom hardware.

The choice for standard over non-standard communication protocols brings some advantages like hardware compatibility and available Commercial Off-The-Shelf (COTS) components. Non-standard systems commonly achieve increased efficiency and improved Quality of Service (QoS).

This chapter presents an overview and a comparison of different wireless communications technologies typically used in mobile robotic systems.

2.1 IEEE 802.11

The IEEE 802.11 standard [34] consists of a series of over-the-air modulation techniques that use the same basic protocol. The most popular are those defined by the IEEE 802.11a, IEEE 802.11b and IEEE 802.11g variants. The goals behind the development of such technology are [24]:

- to deliver services previously found only in wired networks;
- high throughput;
- highly reliable data delivery;
- continuous network connection.

This standard is also commonly known as Wi-Fi and popularized by the Wi-Fi Alliance[6], a trade association that promotes Wireless Local Area Network (WLAN) technology and certifies products that conform to the IEEE 802.11 standards. Since not all IEEE 802.11 compliant devices are submitted for certification to the Wi-Fi Alliance, the lack of the Wi-Fi logo does not necessarily imply that the device is incompatible with other Wi-Fi devices.

The IEEE 802.11 standard has evolved since its inception to enhance speed, security and QoS. The first version was released in 1997 and various Task Groups started working on amendments, identified by letters, following and improving the base standard. In September 2007 a new version was released including the amendments a, b, d, e, g, h, i and j. The current version [34] was released in 2012 and includes 10 new amendments.

Within IEEE 802.11 there are three allowed frequency bands, but the most used are the Industrial, Scientific and Medical (ISM) band at $2.4\,GHz$, and the Unlicensed National Information Infrastructure (U-NII) band at $5\,GHz$. The third frequency band (used by IEEE 802.11y) is only available in the United States of America (USA) and operates in the frequency range $3.65\,GHz - 3.70\,GHz$.

IEEE 802.11b and IEEE 802.11g use the ISM band and may occasionally suffer interference from other equipment emitting in the same band such as microwave ovens, cordless telephones and Bluetooth devices. IEEE 802.11b/g tolerance to interference is achieved using Direct-Sequence Spread Spectrum (DSSS) and Orthogonal Frequency-Division Multiplexing (OFDM) modulation, respectively. IEEE 802.11a uses the U-NII band and OFDM modulation. Table 2.1 summarizes these characteristics.

[6] `http://www.wi-fi.org`

IEEE 802.11 amendment	Release date	Freq (GHz)	Bandwidth (MHz)	Data rate ($Mbps$)	Modulation
(base)	Jun 1997	2.4	22 *	1, 2	DSSS, FHSS
a	Sep 1999	5.0	20	6, 9, 12, 18, 24, 36, 48, 54	OFDM
b	Sep 1999	2.4	22 *	5.5, 11	HR-DSSS
g	Jun 2003	2.4	20	6, 9, 12, 18, 24, 36, 48, 54	OFDM
n	Sep 2009	2.4/5.0	20	7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65, 72.2	OFDM
			40	15, 30, 45, 60, 90, 120, 135, 150	

* Some literature refers to $20MHz$ [3, 25]

Table 2.1: IEEE 802.11 network standards

Each of the ISM and U-NII bands are divided in channels. The ISM band contains 14 channels with $5MHz$ separation, but the availability of channels depends of each regulatory authority. In Europe, channels 1 to 13 are generally available but in the USA the only allowed ones are channels 1 to 11. For the $5GHz$ U-NII band a list of 19 channels are available for Europe and 20 channels for the USA. All the channels in the U-NII band have $20MHz$ separation resulting in non-overlapping channels as opposed to the ISM frequency band, where all channels overlap. As a result only a small set of the available ISM channels can be used simultaneously to avoid overlapping, see Figure 2.1. Security reasons impose the use of Dynamic Frequency Selection (DFS) in U-NII band since some channels use the same frequency range as RADARs. With this method, the end-user is not allowed to fix the communication channel and is the system that, at runtime, automatically chooses the best channel to use.

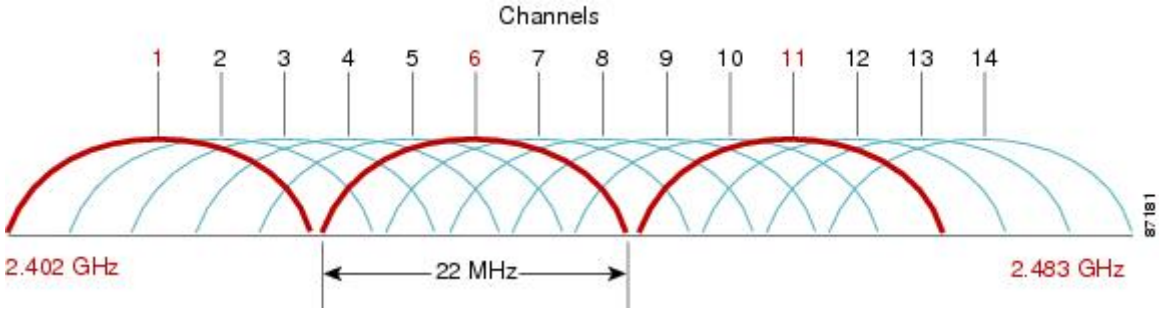


Figure 2.1: IEEE 802.11 $2.4GHz$ ISM band channel overlapping [21]

The basic building block of an IEEE 802.11 network is the Basic Service Set (BSS), which is simply a group of stations that communicate with each other. Stations in the same coverage area can communicate in two different ways, see Figure 2.2: using a) the Independent BSS or, b) the Infrastructure BSS.

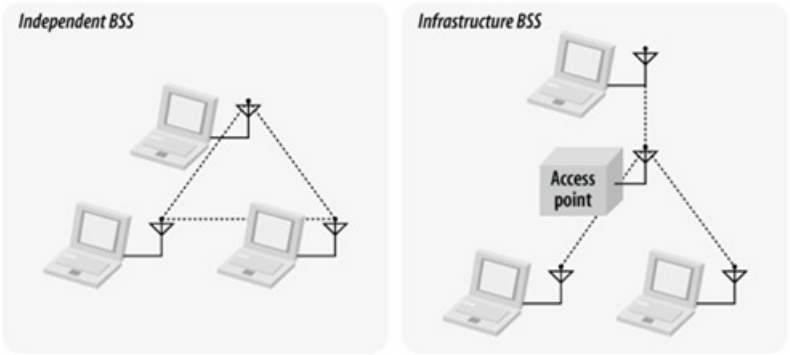


Figure 2.2: Types of IEEE 802.11 networks [31]

In Independent BSS, also known as *ad-hoc* mode, the stations communicate directly with each other, meaning that they must be within direct communication range. To create a minimal Independent BSS, only two stations are necessary. Typically this type of network is set up for a specific purpose and for a short period of time. For example, to exchange information during a meeting or to transfer data between two stations.

Infrastructure BSS is distinguished from Independent BSS by using an AP. The AP mediates all communications, even between stations that could communicate directly. This connection type forces the use of two hops. Firstly, the originating station sends one frame to the AP that, secondly, forwards the frame to the destination station. Since the AP is used as information relay, the coverage area of an Infrastructure BSS is dictated by the coverage of the AP. Conversely, in an Independent BSS, a station can only communicate with those in its direct neighborhood. However, using a routing protocol, the coverage of an Independent BSS can be substantially larger.

In all cases, the most general method to access the medium is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) method. In IEEE 802.11, all the stations in the same network share the same radio frequency (channel) so, only half-duplex communication is possible, meaning that only one station can transmit at a time. Taking advantage of this, and to decrease the final interface costs, only one radio is used at each device, switching between transmission and reception as needed. Without the capacity to listen to what is being transmitted, the sending node cannot detect a collision occurrence. Using CSMA/CA, the node that wishes to send data has to first listen to the channel for a predefined amount

of time to determine whether or not another node is transmitting on the channel within the wireless range. If the channel is *idle*, then the node is allowed to start the transmission. If the channel is *in use*, then the node waits for a random period of time (backoff time) and retries, sensing the channel again. Figure 2.3 shows a simplified flowchart of this process.

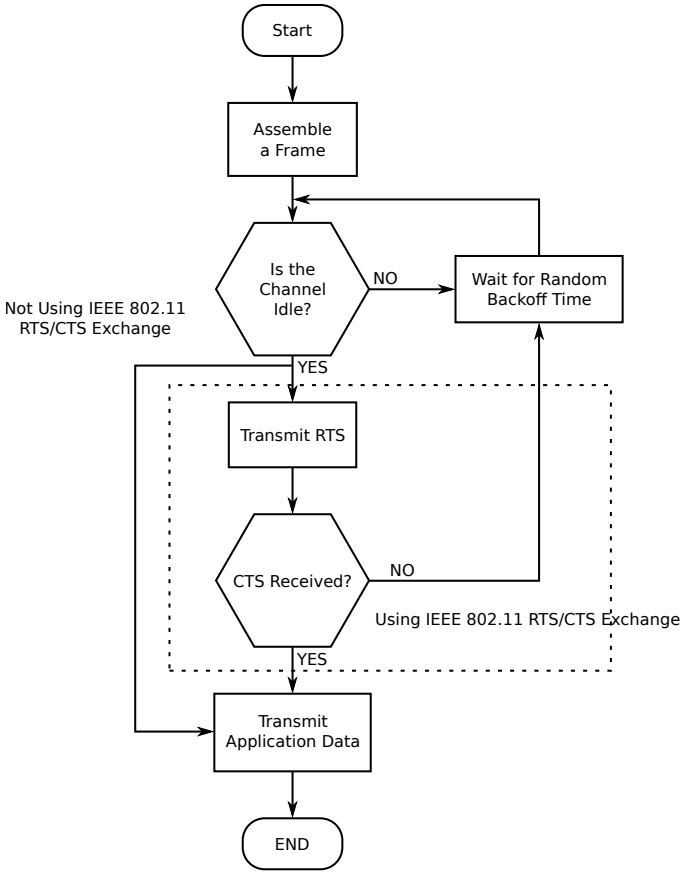


Figure 2.3: Simplified Algorithm of CSMA/CA [56]

Figure 2.3 also includes the Request To Send (RTS) and Clear To Send (CTS) mechanism, optionally used to help solving the hidden node problem. Before starting the transmission of a frame, the sender transmits an RTS packet and the receiver answers with a CTS packet. These packets contain the time needed for the transmission of the original packet, which is encoded in a parameter called Network Allocation Vector (NAV). Thus, all nodes within the range of the sender and the receiver, receive either the RTS, the CTS or both packets and refrain from transmitting for the duration of the main transmission. The hidden node is common in Infrastructure BSS where nodes within the same network not necessarily have to be in the range of each other, since they are considered part of the network if they are in the range of the AP. The hidden node problem occurs when one station wants to transmit

during the transmission of an other station that is too far away to be detected. After sensing the medium, both will be transmitting at the same time and the AP will hear interference, only, and cannot correctly receive any, discarding both packets.

Finally, another issue that is particularly relevant in the scope of this work is power management. In fact, since IEEE 802.11 was developed to support mobility, and mobile stations typically rely on batteries, it is important to reduce the energy consumption of the wireless communications, which is achieved allowing stations to turn off their wireless interfaces when not communicating.

In the particular case of the infrastrutured mode, which we adopt for our work, this turning on and off of the wireless adapters requires synchronizing the stations with the AP so that the stations have their interfaces active when the AP sends them information. On the other hand, stations can send packets at any moment to the AP because it is always active. Then, if needed, the AP buffers any received packets until their destinations have the wireless adapters activated.

This synchronization is achieved with the periodic transmission of a beacon by the AP. Each station activates the wireless interface just before the beacon is transmitted so that it can receive it. The beacon contains a specific field, the Traffic Indication Map (TIM), which informs the existence of buffered data for each destination. The AP then transmits the buffered packets right after the beacon and the stations indicated in the map keep their interfaces active until receiving the packets.

However, broadcast / multicast packets are not signalled in the TIM field because they are sent to multiple destinations. In this case, the AP still buffers these packets, which are transmitted from the stations to the AP as unicast packets, and then disseminates them as broadcast / multicast packets right after a beacon with the signal Delivery Traffic Information Map (DTIM) on. This signal is activated, setting the DTIM interval, every given integer multiple of the beacon period.

Figure 2.4 shows an example of the broadcast / multicast delivery in power management mode. The packets received by the AP from stations 1 and 2 are buffered and transmitted later, right after the beacon message with the DTIM parameter on.

2.2 IEEE 802.15.1

The IEEE 802.15.1 standard [35], commonly known as Bluetooth, is a wireless technology for exchanging data at short distances using the ISM band. Created by Ericsson Mobile Communications in 1994 [12, 39], it was originally conceived as a wireless alternative to

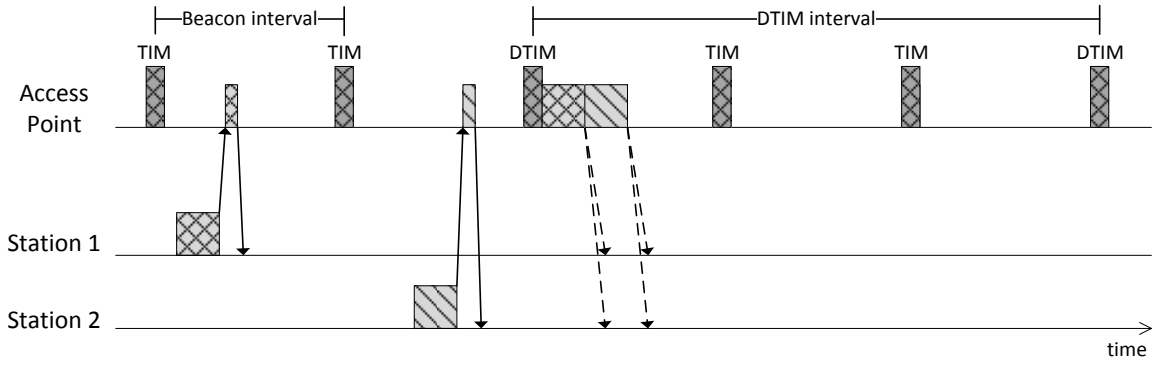
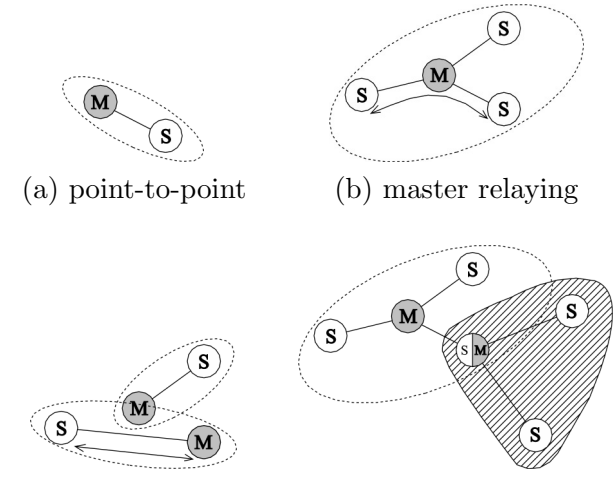


Figure 2.4: IEEE 802.11 power management: multicast and broadcast

RS-232 serial cables. In 1998, a Bluetooth Special Interest Group (SIG)[7] was formed joining Ericsson, Nokia, IBM, Intel and Toshiba. The name Bluetooth comes from a Danish Viking king, from the tenth century, with a discolored tooth, who was famous for uniting Norway and Denmark. Just as Bluetooth technology is designed to unite different business sectors such as computing, mobile phones and automotive industries.

Bluetooth devices use a star connection topology where the central node is named *master* and the rest are named *slaves*, see Figure 2.5. All the communications are managed by the master, i.e., the only way to establish communication between two slaves is through the master. The medium access is controlled by the master who has also to frequently poll the slaves to know if they have data to transmit, typically, in a round-robin fashion.



(a) point-to-point (b) master relaying

(c) separate piconets (d) scatternet

Figure 2.5: Different piconet constellations [43]

[7] http://www.bluetooth.org

A master Bluetooth device can communicate with a maximum of 7 slave devices in a piconet (an ad-hoc network using Bluetooth technology), though not all masters support this limit. In a piconet, there is always one and no more than one master. The devices can switch roles, by agreement, and a slave can become the master. For example, a headset initiating a connection to a phone will necessarily begin as master, since it is the initiator of the connection, but may subsequently prefer to be slave.

When multiple piconets cover the same area, it is possible to form a scatternet, see Figure 2.5.d), in which certain devices simultaneously play the master role in one piconet and the slave role in another. A device may act as slave in more than one piconet, but it is only possible to be master in a single piconet.

The coverage area of a piconet is directly related to the class of the devices in use. Bluetooth defines 3 classes of maximum output power, see Table 2.2. The effective range varies due to propagation conditions, material coverage, production sample variations, antenna configurations and battery conditions.

Class	Max Output Power		Operating range
	mW	dBm	(approximate)
1	100	20	$100m$
2	2.5	4	$10m$
3	1	0	$5m$

Table 2.2: Bluetooth power classes

In order to decrease susceptibility to interference, Bluetooth deploys Frequency Hopping Spread Spectrum (FHSS) modulation. The total bandwidth ($2402\,GHz$ - $2480\,GHz$) is equally divided in 79 channels with a bandwidth of $1\,MHz$ each. Not all the 79 channels are available all over the world due to national regulation constraints. During communication, the system makes 1600 channel hops per second, spreading the communication over all the 79 channels using a long pseudo-random pattern which is generated from the address and clock of the master station in the piconet [27]. Taking advantaged of this method, different piconets will use different hop sequences. When a new slave wants to connect to an existing piconet, it starts by listening the ongoing transmissions and learning the address and clock phase of the master, computing then the hopping sequence.

The first Bluetooth, version 1.0, was released in 1999, and versions 1.1 and 1.2 are from 2001 and 2003 respectively. The PHY and MAC specifications of the Bluetooth releases 1.1 and 1.2 were also rectified as standard IEEE 802.15.1-2002 and IEEE 802.15.1-2005 respectively.

All the following versions are backward compatible with the IEEE 802.15.1 standard that is no longer maintained. Version 2.x provides an Enhanced Data Rate (EDR) feature that uses Phase-Shift Keying (PSK) codding improving the maximum data rate. Version 3.0 and followers provide [90]:

- Improved Speed – The Generic Alternate MAC/PHY (AMP) in Bluetooth high speed enables the creation of ad-hoc Wi-Fi connection between devices when they need to transfer high speed and big sized data;
- Power Optimization – The high speed radio is only enabled when it is necessary, reducing power consumption, which means a longer battery life;
- Lower Latency Rates – Unicast Connection-Less data improves the speed, lowering latency rates, sending small amounts of data more quickly.

Table 2.3 summarizes the Bluetooth technology evolution.

Core Version	Release date	Data rate *Mbps*
1.1	Feb 2001	0.7
1.2	Nov 2003	0.7
2.0 + EDR	Nov 2004	2.1
2.1 + EDR	Jul 2007	2.1
3.0 + HS	Apr 2009	up to 24 *
4.0	June 2010	up to 24 *
4.1	December 2013	up to 24 *

* only with AMP. Bluetooth remains 2.1*Mbps*

Table 2.3: Evolution of the bluetooth data rate

2.3 IEEE 802.15.4

IEEE 802.15.4 is a standard which specifies the PHY and MAC for Low-Rate Wireless Personal Area Network (LR-WPAN) in the range of 10 meters [36]. The initial target was to provide Radio Frequency (RF) communications to applications that require low data rate and long battery life at low cost. IEEE 802.15.4 chip vendors typically sell integrated radios and micro-controllers with flash memory and batteries, resulting in a very small and compact systems, see Figure 2.6.

Figure 2.6: Example of IEEE 802.15.4 board [19]

Some of the characteristics of the IEEE 802.15.4 standard are:

- Over-the-air data rates of $851kbps$, $250kbps$, $100kbps$, $40kbps$, and $20kbps$
- Star or peer-to-peer operation
- Allocated short $16bit$ or extended $64bit$ addresses
- Optional allocation of Guaranteed Time Slots
- CSMA/CA or ALOHA [2] channel access mechanisms
- Fully acknowledged protocol for transfer reliability
- Low power consumption
- Energy Detection
- Link Quality Indication
- 16 channels in the $2.4GHz$ band, 10 channels in the $915MHz$ band, 1 channels in the $868MHz$ band

- optionally: 14 overlapping Chirp Spread Spectrum (CSS) channels in the $2.4\,GHz$ band, 16 channels in Ultra-Wide Band (UWB) band ($500\,MHz$ and $3\,GHz$ to $10\,GHz$), and others special frequencies to be used in China

The standard defines two types of network nodes (see Figure 2.7):

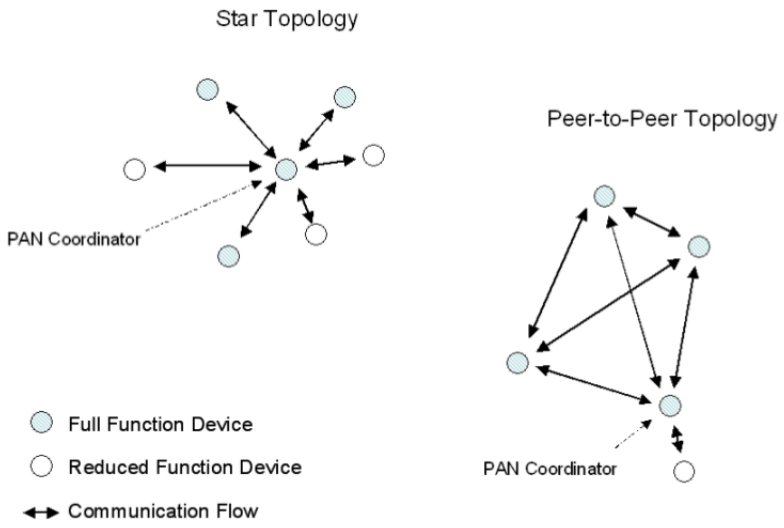


Figure 2.7: IEEE 802.15.4 network topologies [13]

- Full-Function Device (FFD) – It can serve as the coordinator of a Personal Area Network (PAN) just as it may function as a common node. It implements a general model of communication which allows it to talk to any other device: it may also relay messages, in which case it is dubbed a coordinator (PAN coordinator when it is in charge of the whole network).
- Reduced-Function Device (RFD) – These are meant to be extremely simple devices with very modest resource and communication requirements; due to this, they can only communicate with FFD's and can never act as coordinators.

Networks can be built as either peer-to-peer or star/tree networks. However, every network needs at least one FFD to work as the coordinator of the network. Networks are thus formed by groups of devices separated by suitable distances.

Peer-to-peer (or point-to-point) networks can form arbitrary patterns of connections, and their extension is only limited by the distance between each pair of nodes. They are meant to serve as the basis for ad-hoc networks capable of performing self-management and organization. Since the standard does not define a network layer, routing and multi-hop communications are not directly supported.

ZigBee, WirelessHART and MiWi are high level layers, built on top of the IEEE 802.15.4, to further improve the standard according to the application domain.

ZigBee[8] is the most common specification suite of high level communication protocols based on the IEEE 802.15.4 standard, enhancing the standard by adding network and security layers and an application framework. ZigBee supports a large number of interoperable specifications including ZigBee Health Care, ZigBee Home Automation, ZigBee Smart Energy, ZigBee Telecom Services, and the forthcoming ZigBee Building Automation and ZigBee Retail Services.

The WirelessHART[9] specification aims at process monitoring, control and asset management in industry plants. Since 1989 the HART Foundation specifies a communication protocol for smart instruments. WirelessHART technology is a complementary enhancement to the HART Protocol (wired based), providing an additional capability that can benefit both existing and new monitoring and control applications.

MiWi is a proprietary protocol designed by Microchip Technology[10] for use in PIC and dsPIC micro-controllers. The MiWi stack is smaller than ZigBee's (approximately 90% smaller), permitting its use on very limited memory devices. Although the MiWi software is available for free from Microchip website, it can only be used with Microchip micro-controllers.

The main application of the IEEE 802.15.4 standard with the referred high level layers is in Wireless Sensor Network (WSN). These consist of spatially distributed autonomous sensors that monitor physical or environmental variables and cooperatively route their data through the network to a main location, the sink, see Figure 2.8.

2.4 Enhanced / overlay protocols

Beyond the direct use of standard wireless communication technologies, such as those described in the previous sections, there are also other non-standard communication systems that can be used in Robotics. These systems normally derive from the standards, with changes in the MAC. Modifications in MAC are carried out either in the device-driver software or through a firmware modification and are normally the result of research works that seek efficient medium access, energy efficiency, security or QoS improvement or to maximize the throughput.

[8] http://www.zigbee.org
[9] http://www.hartcomm.org
[10] http://www.microchip.com

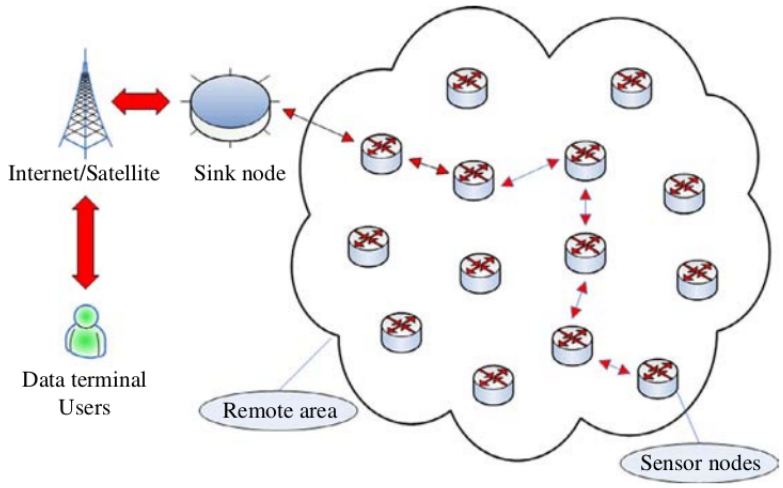


Figure 2.8: Wireless sensor network system [98]

Changes in the PHY are also used, mostly motivated by the need to avoid the frequency interference with other communication protocols that share the same radio band. This is not so common, though, since modified stations cannot interact with standard ones. However, new standards are frequently born this way. A recent one related with the previous technologies is the enhancement of IEEE 802.11 for vehicular communications called Wireless Access in Vehicular Environments (WAVE) and standardized as IEEE 802.11p [95]. However, once this protocol is standardized, then new overlay software protocols have already been proposed to improve certain properties in one direction or another, e.g., the Vehicular Flexible Time-Triggered (V-FTT) protocol for safe communications between vehicles and road side units [62]. IEEE 802.11p is, however, optimized for stations that interact at high speed and thus, is it not particularly relevant for multi-robot teams where interaction speeds are typically much slower.

The space of modified wireless protocols with respect to the standards referred before is too vast to allow an exhaustive enumeration. In fact, there is a myriad of adaptations of the standards, frequently developed in diverse scopes, from wireless sensor networks to mobile ad-hoc networks. Here we refer to just a few that are examples of modifications to the standards referred above, aiming specific properties.

For example, Haghani, Krishnan and Zakhor [40] propose an adaptive carrier-sensing technique to be used on IEEE 802.11, which improves the throughput, using a periodic signal, transmitted by the AP to all the connected stations, providing a way for each station to adjust the Carrier Sense Threshold in order to mitigate the hidden/exposed node problem. This technique requires a modified AP that is able to transmit such periodic signal to all stations. The authors argue that standard stations can still interact with the modified AP

and modified stations, since they simply do not recognize the periodic signal. This technique might be helpful in teams of mobile robots, which, due to mobility, may assume relative positions that favor hidden-nodes.

Balador and Movaghar [9], present an IEEE 802.11 backoff mechanism, which divides the contention window range in various levels based on the history (last three states) of channel status. Instead of resetting the contention window to its initial value at each successful transmission, the authors propose a smooth decrease of the contention window size, since in the majority of cases, the medium continues congested. The modified stations require a new firmware, maintaining the communication capabilities with common standard devices. This is a general overload mitigation technique and as such, can be also helpful in teams of robots, for the case in which their concentration and transmission rate saturate the medium.

Tardioli and Villarroel [97] propose a modified MAC for IEEE 802.11 networks that provides real-time network latency. The protocol works in ad-hoc mode and uses a token-passing approach to query all nodes about the currently ready packet with highest priority. This packet is then allowed to be transmitted. Despite the large overhead implied by one arbitration round per packet, this protocol is an effective solution to carry out fixed-priority packet scheduling on IEEE 802.11 wireless networks. It does not need an infra-structure and tracks the current topology by circulating the token during the arbitration phases.

Another real-time protocol for IEEE 802.11 networks is the one proposed by Costa, Portugal, Vasques and Moraes [18]. This protocol segregates real-time traffic from non-real-time, giving higher priority to the former using shorter interframe spaces and organizing it in a TDMA fashion supported on a beacon that sets the communication round. However, the TDMA structure for the real-time traffic is fixed, which reduces its efficiency for dynamic situations with variable number of active stations as common in teams of robots.

Bartolomeu, Fonseca and Vasques [10] proposed a protocol that concentrates the medium access control in a master node that schedules the network traffic. This can be an effective solution for periodic traffic scheduling but has the downside of having a single point of failure and thus, not being very adequate for teams of robots.

Focusing on a different technology, Golmie and Chevrollier [33] propose two techniques to be applied to IEEE 802.15.1 to improve performance in the presence of IEEE 802.11 interference. One technique is based on controlling the transmitted power and keeping it proportional to the Signal-to-Interference Ratio (SIR) measured at the receiver. The other technique takes advantage of the frequency hopping sequence of IEEE 802.15.1 and uses scheduling with the aim of avoiding the usage of the interference channels.

Flammini *et al* [29] use IEEE 802.15.4 COTS devices with an overlay protocol for real-time communications adopting a hybrid solution that mixes a TDMA protocol on top of the native CSMA/CA. The proposed protocol divides each cycle in a joining period and real-time period. The joining period is where new stations announce their intention to join the network. The remaining time is devoted to real time data communication that occurs by means of TDMA. Once a node is linked to the coordinator, it sleeps for most of the time in power saving mode and periodically wakes up and sends its data. To maintain the clock synchronization and minimize the clock drift a Proportional-Integral (PI) control loop is used. This protocol does not strictly require any change to hardware or firmware of COTS devices but its real-time guarantees are degraded in the presence of uncontrolled traffic, i.e., unaware of the enhanced protocol.

Adopting custom products, Mahlknecht and Durante [58] propose an energy efficient MAC protocol using an extra radio for listening, only. The Wakeup Receiver with ultra-low consumption is permanently active and waiting for incoming requests. This way, complicated algorithms needed to synchronize nodes in traditional WSN are avoided. Authors argue that total energy consumption, adopting this strategy, is decreased, in certain conditions, when compared to common systems. The main disadvantage of the proposal is the use of non commercially available hardware.

2.5 Comparison

Table 2.4 summarizes the main attributes of the described standard protocols.

Standard	IEEE 802.11	IEEE 802.15.1	IEEE 802.15.4
Frequency band	$2.4GHz$; $5GHz$	$2.4GHz$	$868MHz$; $915MHz$; $2.4GHz$
Max bit-rate	$54Mbps$ $150Mbps$ (802.11n)	$2.1Mbps$ $24Mbps$ (AMP)	$851kbps$
Nominal range	$100m$	$10m$	$10m$
Number of channels	14; 20	79	1; 10; 16
Basic cell	BSS	Piconet	Star
Max number of nodes	2007	8	>65000

Table 2.4: Comparison of wireless standards

As shown, the preferred radio band is the ISM $2.4GHz$. The coexistence of multiple protocols and/or other devices like microwave ovens, video transmitters and alarm systems with wireless sensors, cause electromagnetic interference that can reduce the protocol efficiency. To minimize this problem, IEEE 802.15.1 uses frequency hoping while most IEEE 802.11 access point devices provide an automatic channel selection based on channel utilization. Beyond frequency channel selection, both IEEE 802.11 and IEEE 802.15.4 use the CSMA/CA algorithm that verifies the channel occupancy before each transmission. Sikora and Groza [91] examine the mutual effects of the three standards, using real-life equipment, in order to quantify coexistence issues, concluding that the worst scenario is the utilization of IEEE 802.11 and IEEE 802.15.4 at the same frequency, since none of the standards provides dynamic adaptation of the frequency channel.

Other differences between the presented standards are the bit-rate and coverage area. On one hand, IEEE 802.11, that is also known as Ethernet cable replacement, is used world wide, and available on any portable computer on market. Supporting high bit-rates, this standard can be used for large data transfer between computers. It is also the standard that provides the higher area coverage, providing more freedom of mobility. On the other hand, the IEEE 802.15.1 and IEEE 802.15.4 are both classified as Wireless Personal Area Network (WPAN), and their main focus is to provide communication to devices with battery constraints. IEEE 802.15.1 can be used as cable replacement to connect a headset to a mobile phone, for example, and IEEE 802.15.4 is mostly known for use on WSN providing a cable free way to monitor the environment or process variables in large plants.

Lee, Su and Shen [53] present a complete comparison between the above standards, including metrics such as transmission time, data coding efficiency, complexity and power consumption. The authors do not draw any conclusion regarding the best standard, arguing that the suitability of network standards is greatly influenced by practical applications beyond other factors such as communications reliability, roaming capability, recovery mechanism, device price and installation costs.

For the specific target of supporting cooperation among mobile robots, we believe that all of the mentioned standards can be used. Again, the choice for one specific communication protocol must be carried out considering the requirements of the specific application case.

2.6 Summary

In this chapter we presented an overview of some of the actual wireless technologies that can be used in multi-robot systems. The standard protocols have the advantage of compatibility and availability over the non-standard protocols that commonly have as main advantage the achievement of better efficiency and/or QoS.

Considering our objective of supporting multi-robot systems, the typical technologies they use and the particular interest on reusable solutions, we chose the IEEE 802.11 standard that presents an extended coverage and is actually built-in recent laptops and similar electronic devices.

Chapter 3

Collaborative technologies for mobile robotic teams

Multi-robot system or *robotic team* are common names used to designate a group of robots that work together, in cooperation to carry out tasks in an efficient and suitable manner. To execute cooperative behaviors, autonomous robots need to access remote information, typically held by other robots in the team, which requires the use of a wireless communication infrastructure as discussed in chapter 2.

However, building consistent cooperative robotic applications goes well beyond the development of autonomous robots and wireless communications and is further complicated by (a) the existence of several independent robots, (b) possibly using different hardware and (c) with different perceptions and perspectives of the operational environment.

The concept of middleware was introduced to facilitate the task of developing cooperative distributed robotic applications. Many interesting definitions of middleware exist, all centered on sets of tools, data and methods that facilitate using networked resources and services. Klingenstein [47] gives a curious definition:

> *"Middleware is the intersection of the stuff that network engineers don't want to do, with the stuff that application developers don't want to do"*

In The Free Dictionary[11], middleware is defined as:

> *"Software that serves as an intermediary between systems software and an application"*

[11] `http://www.thefreedictionary.com`

In Robotics the middleware is an abstraction layer that resides between the operating system and the applications software, designed to handle the heterogeneity of the hardware, improving application software quality, simplifying application software design and reducing development costs and time. The middleware can also provide the glue logic to connect application components, allowing their independent design and development, as well as their integration with other existing components. Furthermore, if a component needs to be modified or improved, it is only needed to replace the old one with the new one and thus, the middleware also plays an important role in improving the maintainability of the distributed application. Finally, the middleware must also provide a mechanism to exchange data among the physical or logical entities that compose the application, with appropriate semantic and meta-data information (e.g., age), to facilitate sensor fusion and/or cooperative actions.

According to [64, 65] the requirements that a middleware for multi-robot systems must fulfill include:

1. **Simplify the development process** – Providing developers with high-level abstractions using simplified interfaces facilitating software integration and reuse;

2. **Support communications and interoperability** – Robotic modules can be developed by different manufacturers/teams and the communication and interoperability between such modules should be efficient and simple;

3. **Provide efficient utilization of available resources** – A robot can be a complex system with different types of sensors, multiple processors and/or microprocessors, one or more interconnection networks, among other resources. The middleware should provide an efficient way to interconnect all the resources for different application requirements without generating significant overhead in any of those resources;

4. **Provide heterogeneity abstractions** – Robots commonly have heterogeneous hardware and software modules, that should be interconnected by means of abstraction layers that hide the complexity of the low-level communication and the diversity of modules;

5. **Support integration with other systems** – Different robots, from different vendors or with different resources can be part of a team. To facilitate the integration among the different robots, the communication among them should be done in an abstract way and in real-time;

6. **Offer often-needed robot services** – Common existing algorithms are often rewritten due to changes in hardware or to adapt to new distributed applications or to new operating systems, or even because developers change. By using a middleware to develop such algorithms, those needs for rewriting can be moved from the application side to the low-level platform software. Thus, the application software remains more stable and reusability is improved;

7. **Provide automatic resource discovery and configuration** – Dynamic systems like mobile robots can be available/unavailable to the other robots in the team and thus, the team must adapt, at run-time, to the changing configuration and available resources;

8. **Support embedded components and low-resource-devices** – Sometimes it is necessary to interact with embedded devices that have limitations like power, memory, operating system functionalities and limited connectivity. In these cases, the middleware should also provide special functions to interact with such devices.

This chapter presents a brief survey of existing distribution middleware technologies and discusses their suitability for multi-robot applications as well as their compliance with the requirements defined above.

3.1 CORBA

The CORBA is a standard proposed by a consortium of companies called Object Management Group (OMG)[12], founded in 1989, that enables the inter-operation of objects running on different platforms, different machines and written in different programming languages. The OMG does not make software and only creates the specifications. The first version of the CORBA standard was released in August 1991 and the major version (v.3) is dated from June 2002. Since 2002 some revisions were made, resulting in versions: 3.1 from January 2008, 3.2 from November 2011 and the most current 3.3 from November 2012 [77].

The CORBA run-time system consists of an Object Request Broker (ORB), which routes calls from one object to another and returns the results. An ORB handles all the invocation, such as finding the target object and corresponding arguments. All object invocations go through the ORB, even if the target object is in the same process.

[12] `http://www.omg.org`

All objects are accessed using interfaces, which are specified in a CORBA-specific Interface Description Language (IDL). The IDL provides support for exceptions, modules and multiple interface inheritance. Aside from an ORB, every CORBA implementation also includes an IDL compiler to generate language-specific proxies. These language-specific proxies are used to issue and deliver ORB and are known as stubs if running on the client, and skeletons if running on the server.

Among others, CORBA has a naming service that allows clients to look up server objects by name.

Figure 3.1 illustrates the high-level paradigm for remote inter-process communications using CORBA.

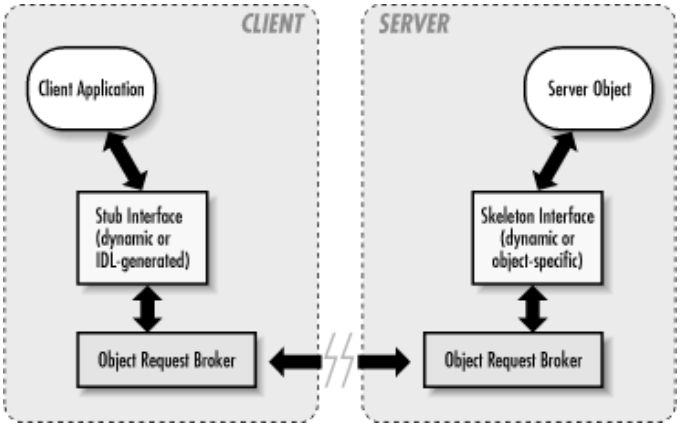


Figure 3.1: Basic CORBA architecture [30]

To adapt the CORBA standard to particular specific application domains, several groups were created. In the Robotics domain the most relevant specifications are:

- **Real Time CORBA (RT-CORBA)** – Formed with the goal of supporting real-time applications, particularly those in which there are end-to-end timing constraints across a distributed system [102]. The RT-CORBA is defined as an extension to CORBA and the last release is dated from January 2005 [74];

- **Robotic Technology Component (RTC)** – Specifies a component model that meets the requirements of robotic systems, extending the general propose CORBA models, focusing on structural and behavioral features of robotic applications [76];

- **CORBA for *embedded* (CORBA/e)** – Specifies a CORBA subset specially designed for systems with limited resources [75]. For small devices, CORBA is too large to meet size and performance constraints. CORBA/e define two profiles for use with embedded devices: Compact Profile, formerly known as Minimum CORBA and, Micro Profile for devices with severely constrained resources.

There are multiple implementations of the OMG CORBA standard and some of them with special focus on robotic applications, as The ACE ORB (TAO) and Real-Time Middleware (RT-Middleware), which we will briefly describe next.

TAO is a CORBA implementation designed for high performance and real-time application [32, 88]. The project was funded by the DARPA Quorum program, National Science Foundation and several industrial sponsors. The main motivations for development was:

1. *Empirical determination of the capabilities needed to enable RT-CORBA ORBs to support mission-critical Distributed Real-time and Embedded systems with hard and soft QoS requirements.*
2. *Combine the strategies for real-time Input/output (I/O) subsystem architectures and optimizations with ORBs to provide vertically-integrated ORB end systems that can support end-to-end throughput, latency, jitter, and dependability QoS requirements.*
3. *Capture and document the key design patterns and optimization principle patterns necessary to develop standards-compliant, portable, and extensible QoS-enabled ORBs.*
4. *Provide a high-quality, freely available, open-source CORBA-compliant middleware platform that can be used effectively by researchers and developers.*
5. *Guide various CORBA-related standardization efforts within the OMG, in particular, the RT-CORBA specification.*

TAO has played an important role in influencing key features in the OMG's RT-CORBA specification, particularly its capabilities for explicit binding and portable synchronizers.

TAO has also facilitated the application of CORBA to Mobile Robotics. For example, Middleware for Robots (MIRO) is a distributed object oriented middleware for mobile robot control, based on TAO technology, developed at the University of Ulm, Germany [23, 99]. To overcome location and programming language dependencies, all sensor and actuator services export their interfaces as network transparent CORBA objects, which can be addressed from any language and platform for which language bindings and CORBA implementations exist.

RT-Middleware is another example of a Robotics-oriented CORBA-based middleware, developed by the National Institute of Advanced Industrial Science Technology (AIST), Japan [6]. The main goal of this middleware is to support efficient development of robot systems using the Unified Modeling Language (UML) so that their functional parts can be connected, in a modular way, at the application software level. This work also made several contributions to the OMG RT-CORBA specification.

In RT-Middleware all the components such as motors, sensors, cameras or even software algorithms are regarded as Real-Time (RT) functional elements. The software component of each RT functional element is called Real-Time Component (RT-Component). These components have interfaces (also called ports) to communicate with other components and to exchange data. An RT system is constructed by connecting the ports of multiple components as an aggregation of RT-Component functions.

3.2 DDS

The DDS for real-time systems, also from OMG, is a specification of a Publish-Subscriber (PS) middleware for distributed systems created in response to the need to standardize a data-centric PS programming model for distributed systems [79, 80].

A PS model connects information producers (the publishers) with information consumers (the subscribers). The overall distributed application (the PS system) is composed of processes, where each could be running in a separate address space or even on different computers. Each process can be simultaneously a publisher and a subscriber of information.

In DDS the association of a DataWriter object (representing a publication) with a DataReader object (representing the subscription) is done by means of the Topic, see Figure 3.2. Each topic has associated a name (unique in the system), a data type, and a desired QoS related to the data itself. The type definition provides enough information for the service to manipulate the data (for example serialize it into a network-format for transmission). The definition can be done by means of a textual language or by means of an operational "plugin" that provides the necessary methods.

The information transferred by DDS communications can be further classified into:

- **Signals** – Represent data that is continuously changing (such as the readings of a sensor). Signals can often be sent using best-effort transmission approaches;
- **Streams** – Represent snapshots of the value of a data-object that must be interpreted in the context of previous snapshots. Streams often need to be sent reliably;

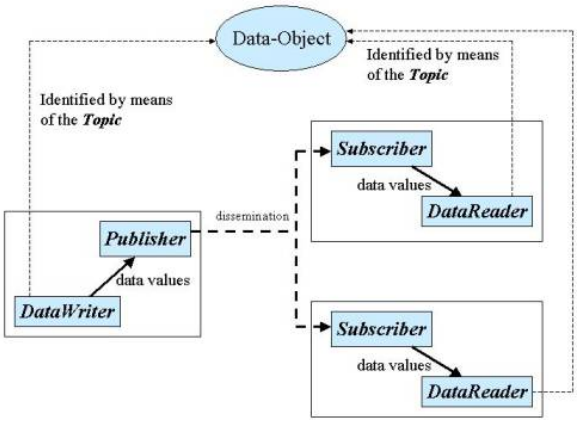


Figure 3.2: Simple DDS conceptual flowchart [37]

- **States** – Represent the state of a set of objects (or systems) meaning the most recent value of a set of data attributes (or data structures). States are transmitted upon change of their own value, i.e., upon event occurrence, and must be transmitted reliably.

The state of an object does not necessarily change with any fixed period. In fact, events are asynchronous and cannot be foreseen. Fast changes may be followed by long intervals without change. Consumers of "state data" (events) are typically interested in the most recent state. However, as the state may not change for a long time, the middleware may need to ensure that the current state is delivered reliably. In other words, if a value is missed, then it is not always acceptable to wait until the value changes again, i.e., the next event occurs.

The goal of the DDS specification is to facilitate the efficient transmission of data in a distributed system. Participants using DDS can "read" and "write" data efficiently and in a similar way to the one used to read/write local variables. Underneath, the DDS middleware will transmit the data so that each reading participant can access the "most-current" values. The service creates a global "data space" that any participant can read from and write to, and also creates a name space to allow participants to find and share objects.

As initially stated, DDS targets for real-time systems. The Application Programming Interface (API) and QoS support are chosen to balance predictable behavior and implementation efficiency/performance [79]. The DDS specification describes two levels of interfaces:

- A lower Data-Centric Publish-Subscribe (DCPS) level, that is targeted towards the efficient delivery of information to each recipient;
- An optional higher Data-Local Reconstruction Layer (DLRL) level, which allows for a simpler integration into the application layer.

DDS is currently at version 1.2, released in January 2007 [37]. It is available from many different vendors with support for multiple programming languages, e.g., Ada, C, C++, C#, and Java.

Given that DDS implementations from distinct vendors must be able to inter-operate, a DDS Interoperability (DDSI) - Real-Time Publish Subscribe (RTPS) communication protocol was published. RTPS has the last specification revision in November 2010 [38].

RTPS was specifically developed to support the unique requirements of data-distribution systems within industrial automation, and contribute to a standard publish-subscribe wire-protocol that closely matched that of DDS. As a result, there is significant synergy between DDS and the RTPS wire-protocol designed to run over multicast and connectionless best-effort transport layers such as User Datagram Protocol (UDP)/Internet Protocol (IP). The main features of the RTPS protocol include [38]:

- **Performance and Quality-of-Service support** – to enable best-effort and reliable publish-subscribe communications for real-time applications over standard IP networks;
- **Fault tolerance** – to allow the creation of networks without single points of failure;
- **Extensibility** – to allow the protocol to be extended and enhanced with new services without breaking backward compatibility and interoperability;
- **Plug-and-play connectivity** – so that new applications and services are automatically discovered and applications can join and leave the network at any time without the need for cold reconfiguration (with the system halted);
- **Configurability** – to allow balancing the requirements for reliability and timeliness for each data delivery;
- **Modularity** – to allow simple devices to implement a subset of the protocol and still participate in the network;
- **Scalability** – to enable systems to potentially scale to very large networks;
- **Type-safety** – to prevent application programming errors from compromising the operation of remote nodes.

The RTPS specification defines the message formats, interpretation, and usage scenarios that underly all messages exchanged by applications that use the protocol.

OCERA Real-Time Ethernet (ORTE) [94] is an open source implementation of RTPS. RTI Connext DDS [83], formely known as NDDS, is a commercial DDS implementation from the RTI company[13].

ORTE was developed at Czech Technical University from Prague, Czech Republic and is one of the communication components used by the Open Components for Embedded Real-Time Applications (OCERA) project[14]. ORTE is available as an independent package and is used by some robotic teams [22, 26, 44].

3.3 ICE

Internet Communications Engine (ICE)[15] is an object-oriented middleware that provides object-oriented Remote Procedure Call (RPC), grid computing and PS functionality [42, 103]. Developed by ZeroC and dual-licensed under the GNU General Public License (GPL) and a proprietary license.

ICE allows to write distributed applications in C++, Java, C# (and other .NET languages, such as Visual Basic), Python, Ruby, PHP, and ActionScript. ICE is currently available in three products:

- ICE – The main product for use in mainstream platforms, such as Windows and Linux;
- ICE-E – An implementation of ICE for resource-constrained systems, with support for C++ and embedded operating systems (Windows CE, Linux). ICE-E does not include any services. However, ICE-E applications can use most of the services provided by ICE, so mobile devices can be seamlessly integrated into a distributed system;
- ICE Touch – A specific product for Apple devices (iPhone and iPod touch). ICE Touch includes an Objective-C implementation with supports for iPhone OS and provides full access to the Cocoa framework for developing graphical applications for Mac OS X.

With these different versions, ICE allows the use of heterogeneous distributed systems, with distinct memory and processing capacity, over multiple operating systems and programming languages, see Figure 3.3.

ICE supports synchronous and asynchronous calls, allows messages to be batched for efficiency, and permits sophisticated control of threads and resource allocation. Multiple instances of a server can be deployed on different machines, with transparent fail-over if

[13] `http://www.rti.com`
[14] `http://www.ocera.org`
[15] `http://www.zeroc.com`

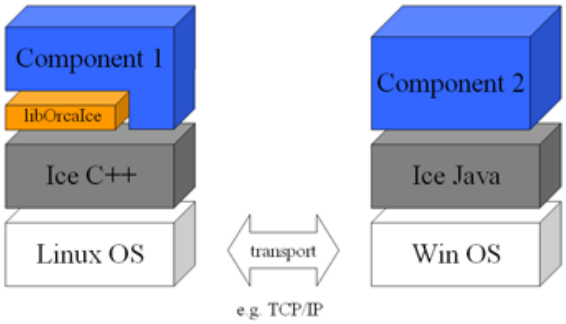


Figure 3.3: Example of application of ICE middleware [59]

a machine crashes or is disconnected from the network. This not only makes applications resilient against failures, but also increases performance because ICE allows to balance the load of a distributed system over several servers.

At the network level, ICE uses a protocol that minimizes the bandwidth consumption.

Open Robot Controller Architecture (ORCA) is an open-source framework based on ICE for developing component-based robotic systems. It provides the means for defining and developing the building-blocks which can be pieced together to form arbitrarily complex robotic systems, from single vehicles to distributed sensor networks [59, 60].

The project's main goal is to promote software reuse in Robotics. The ORCA developers defined the following middleware requirements:

1. *Open-source;*
2. *Distributed under a license which allows use in commercial applications;*
3. *Modular;*
4. *Distributed with a repository of tested and documented modules.*
5. *General, flexible and extensible;*
6. *Sufficiently robust, high-performance and full-featured for use in commercial applications;*
7. *Sufficiently simple for experimentation in university research environments.*

To satisfy these requirements, ORCA:

- Adopts a Component-Based Software Engineering approach without applying any additional constraints (requirements 3, 5);

- Uses a commercial open-source library for communication and interface definition – ICE (requirements 5, 6);
- Provides tools to simplify component development but make them strictly optional to maintain full access to the underlying communication engine and services (requirements 6, 7);
- Uses cross-platform development tools (requirement 5).

3.4 SOAP and ROS

The Simple Object Access Protocol (SOAP)[16], defined by the World Wide Web Consortium (W3C), provides a simple mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using Extensible Markup Language (XML) [20]. Initially named as XML-RPC protocol in 1998 has evolved to SOAP that is currently at version 1.2, released in April 2007. XML-RPC could be viewed as a subset of SOAP, since it lacks some descriptors.

SOAP itself does not define any application semantics such as a programming model or implementation specific semantics. Rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC.

SOAP consists of three parts:

- **Envelope** – Defines an overall framework for expressing what is in a message; who should deal with it, and whether it is optional or mandatory;
- **Encoding rules** – Defines a serialization mechanism that can be used to exchange instances of application-defined datatypes;
- **RPC representation** – Defines a convention that can be used to represent remote procedure calls and responses.

The Listing 3.1 shows an example of a XML-RPC message.

The most well known implementation of XML-RPC in Robotics is the Robot Operating System (ROS)[17]. It is a middleware for robot software development, providing operating system-like functionality on heterogeneous computers. ROS was originally developed in the

[16] `http://www.w3.org/TR/soap`
[17] `http://www.ros.org`

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>Hello World.</string></value>
    </param>
  </params>
</methodResponse>
```

Listing 3.1: Example of a XML-RPC message

mid-2000s by the Stanford Artificial Intelligence Laboratory within the Stanford AI Robot (STAIR) project. After 2007, the development continued at Willow Garage for their robot PR2, but quickly was adopted to be used in other robots. Under a Berkeley Software Distribution (BSD) open-source license, ROS gradually has become a widely-used platform in the robotics research community.

ROS provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It is based on a graph architecture where processing takes place in nodes. ROS supports both PS and client–server models.

ROS has two basic sides: The middleware side as described above and the framework side – `ros-pkg`, a suite of user contributed packages that implement functionalities such as simultaneous localization and mapping, planning, perception, simulation, etc.

Taking the best from different middlewares and frameworks, the authors tried to create a complete solution with gateways to allow interconnection with existing hardware and software. Several commercial robots and sensors are well supported.

ROS is released under the terms of the BSD license, and is open source software. The `ros-pkg` contributed packages are licensed under a variety of open source licenses.

As stated by Quigley *et al* [81], the philosophical goal of ROS is to be:

- **Peer-to-peer** – A system built on ROS consists of a number of processes, possibly on different hosts, connected at runtime in a peer-to-peer topology. The lookup mechanism to allow processes to find each other at runtime is called *name service* or *master*;

- **Tools-based** – In an effort to manage the complexity of ROS, the authors opted for a microkernel design, where a large number of small tools are used to build and run the various components, avoiding a monolithic development and runtime

environment. Examples of these tools are: navigate the source code tree; get and set configuration parameters; visualize the peer-to-peer connection topology, measure bandwidth utilization, etc;

- **Multi-lingual** – ROS was designed to be language-neutral. The ROS specification is at the messaging layer and the peer-to-peer connection, negotiation and configuration occurs in XML-RPC, for which reasonable implementations exist in most major programming languages. To better follow the conventions of each language, the developers have opted to implement ROS natively in each language;

- **Thin** – To better re-use algorithms or drivers outside the framework they must be developed as standalone libraries that have no dependencies on ROS. In this way, unit testing is often far easier, as standalone test programs can be written to exercise various features of the library. ROS actually re-uses code from numerous open-source projects: Player project for drivers, navigation and simulators; OpenCV for vision algoritms; OpenRAVE for planning algoritms; among others;

- **Free and Open-Source** – The full source code of ROS is publicly available. Authors believe this to be critical to facilitate debugging at all levels of the software stack.

3.5 Comparison

As expected, all the mentioned middleware standards and respective implementations clearly satisfy the first four requirements proposed by [64, 65] presented in the beginning of this chapter, namely: (1) Simplify the development process; (2) Support communications and interoperability; (3) Provide efficient utilization of available resources and (4) Provide heterogeneity abstractions.

The last four requirements are not fully addressed:

- ROS does not *support integration with other systems (5)* since it lacks support for Microsoft Windows Operative System;

- ORTE does not *offer often-needed robot services (6)* since it focuses on communication, only;

- MIRO does not *provide automatic resource discovery and configuration (7)*;

- CORBA and ICE are the only standards that have specific references to *support embedded components and low-resource-devices (8)* using a specific micro kernel.

For the network utilization, the ROS middleware could theoretically lead to slower communications because of the verbosity of the XML, when compared with the other standards.

3.6 Summary

This chapter presented a discussion over the currently most used middlewares in the Robotics domain.

As stated in Chapter 2 a wireless network is mandatory for mobile multi-robot systems. However, none of the discussed standards/implementations have a specific focus on the wireless network constraints.

As a conclusion of this chapter, we refer to [64, 66] which present a specific comparison of networked robots systems. They observe that, "*although the middleware solution is very useful, it is difficult to have one middleware platform that can offer all the required features and functionalities for collaborative robotic systems*" and then conclude "*therefore, it is more practical to consider several approaches suitable for different requirements, while maintaining some guidelines to facilitate reuse, interoperability and integration*".

Chapter 4

RoboCup MSL communications: problems and requirements

The RoboCup[18] [45, 46] MSL has been an effective test-bed for Cooperative Robotics. In fact, beyond all the issues associated with the construction of robots for operation in harsh conditions, each team needs to develop coordinated behaviors to beat the opponent team. This cooperation is becoming more sophisticated involving the communication of team mates positions, fusion of ball position information, dynamic role assignment, formations and ball passes, among others. Most of the participating teams develop cooperative behaviors on top of a middleware, as presented in Chapter 3, that facilitates information exchange among the team members. In turn, such middleware relies on a wireless communication protocol, namely IEEE 802.11, as presented in Chapter 2.

Despite its importance the wireless communication is known to be less reliable than its wired counterpart with significantly higher bit-error rates, to have limited and variable bandwidth and to be open to the access by other stations not involved in the team, among other undesired phenomena [101]. Moreover, the wireless channel must be shared by both teams involved in a game, thus becoming a critical shared resource.

This chapter presents an analysis of the use of the wireless channel by several MSL teams during a RoboCup event. It shows a substantial difference between teams, with some of them making a parsimonious use of the channel while others use substantial slices of the available bandwidth. Few teams transmit in a sparse periodic fashion and others send bursts of data within very short time intervals. The patterns of transmission depend to a large extent on the middleware layer that manages the exchange of information.

[18] `http://www.robocup.org`

In this chapter we will address the problem of information sharing in the specific case of the RoboCup MSL. We identify recurring problems and misconceptions and define the requirements for an adequate solution at the level of the communications protocol and distribution middleware. We believe that such solution can also be applied to a broad class of situations among which the RoboCup MSL is just one example.

4.1 Wireless communication within the MSL

Since several years, the MSL rules stipulate that the wireless technology to be used is IEEE 802.11a/b. The more popular IEEE 802.11g technology is not allowed simply because it uses the same band as IEEE 802.11b but with fewer non interference frequency channels, despite with larger bandwidth. This increases the difficulty in channel planning and assignment per competition area to minimize cross-interference, as discussed in section 2.1. Generally, one channel is assigned to one competition field and both teams playing therein must share it. An attempt is always made to assign non-interfering channels to neighboring fields. Moreover, the communication must be infra-structured, i.e., using access points, while direct ad-hoc communication is not allowed.

Based on this study, some limitations were introduced in the rules of the MSL RoboCup 2009 edition. The bandwidth allowed to each team was limited to $2.2 Mbps$ until recently[19] and the utilization of broadcasts is not allowed. Briefly, the MSL rules, concerning the wireless communication, currently stipulate:

- IEEE 802.11a/b technology
- Infra-structured mode (through AP)
- Single a plus single b channels per game (each shared by both teams)
- IP v4 addressing within predefined networks
- Only unicasts/multicasts (broadcasts are forbidden)
- Up to 2.2Mbps bandwidth utilization per team (2009-2013)

The bandwidth limitation was calculated considering the lower bandwidth technology IEEE 802.11b ($11 Mbps$), which is still used by some teams due to national regulations. This is applied to both types of network, either 'b' and 'a' for fairness reasons. Also, the traffic

[19] In 2013 this limitation was removed simply because of organizational difficulties in verifying it and because abusive behavior had not been detected for some time.

is replicated on both networks, meaning that if during a game each team uses a different wireless technology, all the traffic is available at IEEE 802.11a and IEEE 802.11b, again for fairness reasons.

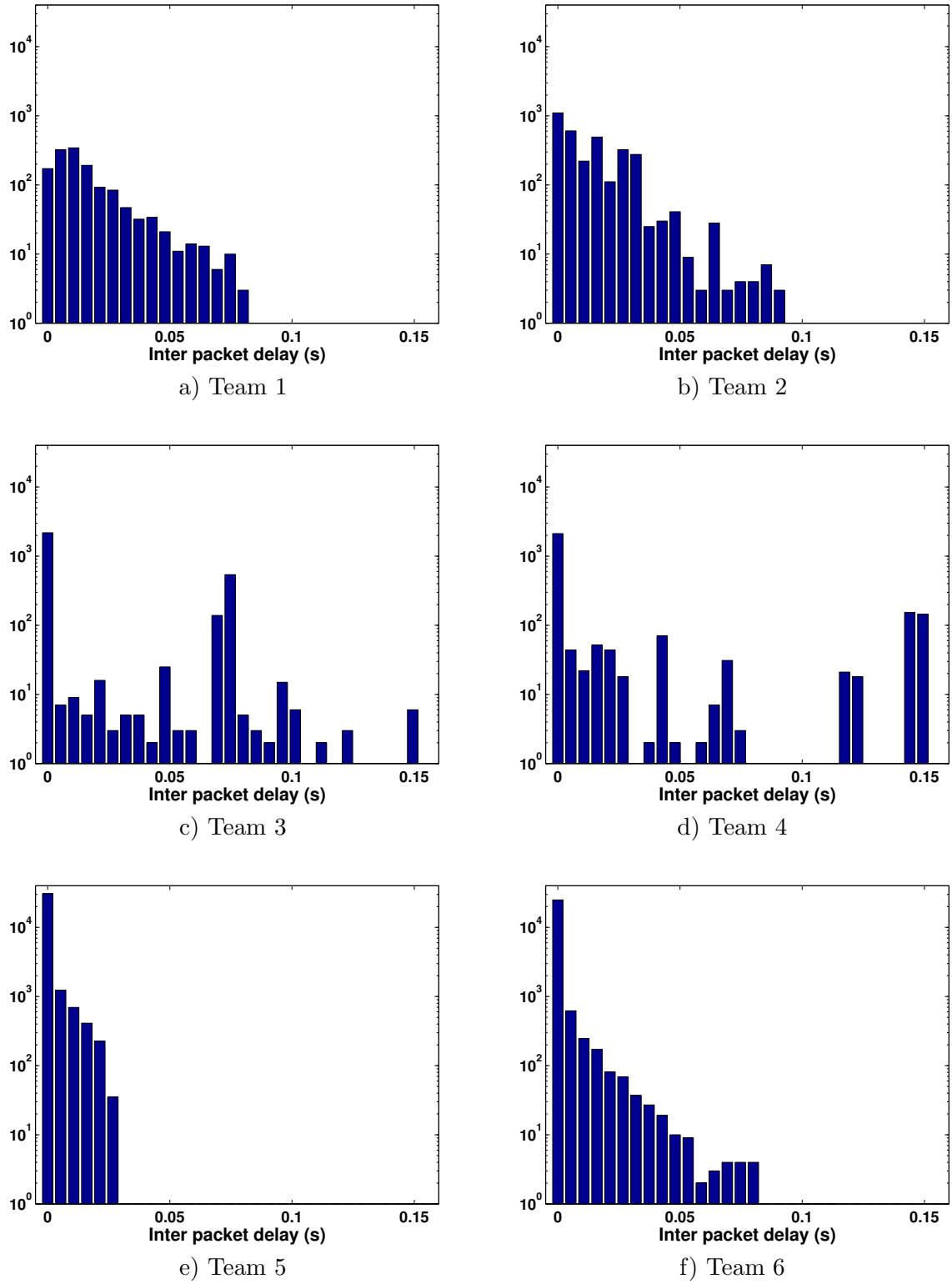
4.2 Logs from the MSL RoboCup

In order to gather information on how teams use the wireless channel, the communications were monitored during several games of RoboCup 2008, in Suzhou - China, a RoboCup championship won by our team, CAMBADA. We used a laptop with a wireless adapter configured in monitor mode, which disables filtering and allows receiving all IEEE 802.11 packets that arrive at its antenna for a predefined channel. The monitoring software was the Wireshark network protocol analyzer and six teams were monitored, during periods of approximately 1 minute, randomly taken during the third round-robin games. In all these games, all communications took place using IEEE 802.11a, but the effective bit-rates achieved during the competitions varied widely between $6 Mbps$ and $54 Mbps$ with an approximate average of $36 Mbps$.

Figure 4.1 shows a set of histograms concerning the distribution of the inter packet intervals related to each team considering the transmissions of all its members as they are effectively transmitted in the wireless medium. One can clearly identify three different classes.

One class includes teams 1 and 2, that do some level of traffic spread in the time domain, exhibiting inter-packet intervals that extend up to approximately $80 ms$. Team 1 uses multicast packets to share information in a producer-consumer fashion. On the other hand, team 2 uses unicasts, with the robots exchanging information between them in pairs.

Another class, formed by teams 3 and 4, shows a clear dual mode operation with many packets sent in sequence but others sent with longer well defined intervals. Looking in more detail to their logs, it was possible to identify that all robots of team 3 transmit periodically and synchronized, with all robots transmitting in sequence and then waiting for a period of approximately $75 ms$. This team used IP broadcast frames to exchange information in a producer-consumer fashion. On the other hand, team 4 uses a middleware probably based on a centralized Blackboard that resides in one particular station to which all robots send their sensing data periodically, approximately every $150 ms$, but often faster. Then, such station carries out some computation, probably sensor fusion, and delivers the result back to the nodes in unicast packets sent in sequence, thus generating a peak of packets sent within a very short interval.



a) Team 1 b) Team 2

c) Team 3 d) Team 4

e) Team 5 f) Team 6

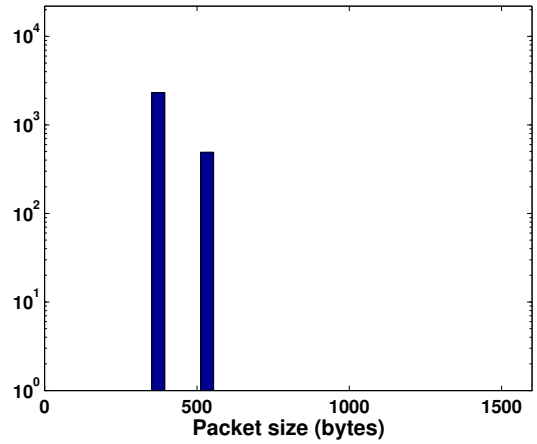
Figure 4.1: Histograms of inter-packet intervals for each team (s)

Finally, a third class includes teams 5 and 6, that sent their traffic in an almost continuous fashion, with very short intervals, leading to numbers of packets that are an order of magnitude higher, with too high bandwidth utilization levels, when compared to the other classes.

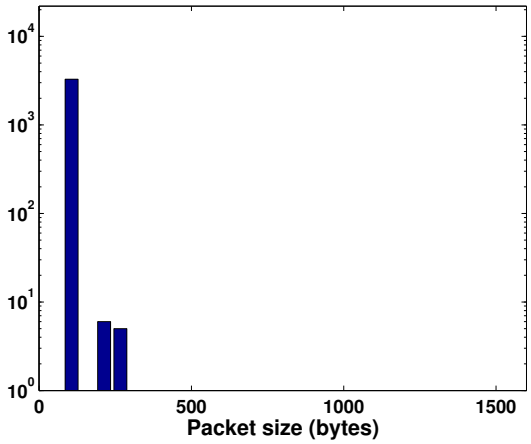
Figure 4.2 shows the histograms of the packet sizes used by each team, in bytes. Clearly two situations arise. Teams 1 through 4 use mainly fixed size packets, in some cases with two different sizes, team 1 with average size packets and teams 2, 3 and 4 with relatively small packet sizes. In turn, teams 5 and 6, use a wide variability of packet lengths, with significant use of large ($1.5KB$) packets. These teams were the only ones sending bursts of information, too. We detected bursts of up to six consecutive $1.5KB$ packets in the case of team 5 and up to twelve consecutive $1.5KB$ packets in the case of team 6. In the IEEE 802.11a technology, these twelve consecutive packets, at the maximum bit rate of $54Mbps$, cause an interference of approximately $10ms$, but in IEEE 802.11b technology these bursts would imply about $50ms$ delays, at the best possible conditions, i.e, without collisions or transmissions errors.

Table 4.1 shows a summary of the main traffic statistics of the monitored teams, covering inter-packet interval in milliseconds, packet size in bytes, burst size in number of consecutive $1.5KB$ packets, total number of bytes transmitted in the monitoring interval and respective approximate utilization in IEEE 802.11a/b channels. The traffic classes that were identified in the analysis of the histograms are naturally reflected in this table but the information on the approximate bandwidth utilization of the IEEE 802.11a/b channels reveals the huge variations in channel utilization. It is curious to see that team 5 was already using approximately 25% of the IEEE 802.11a channel, which corresponds to about 125% of the width of an IEEE 802.11b channel. The figures for team 6 are slightly better but still revealing a substantial channel overuse. The other teams use significantly lower bandwidths, near 2 orders of magnitude less, which allows them to play without problems among each other using any of the two kinds of channels. One curious detail is the fact that team 6 was using 11 different computers, substantially more that the maximum of 6 robots plus one remote station.

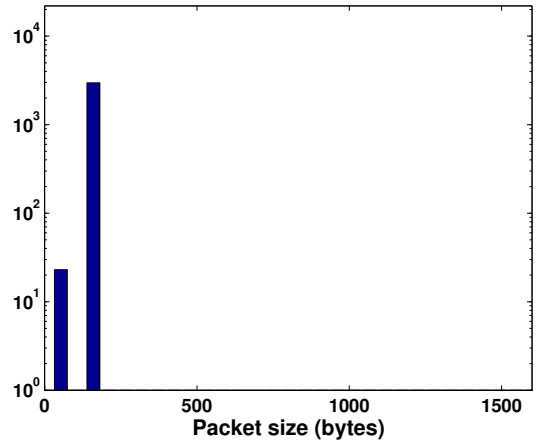
Figure 4.3 shows the impact that different opponents can have on the timeliness of the transmissions of a robot. In this particular case we used robot 1 of team 2 (any other robots yielded similar results) in two games, one against team 1 that makes a relatively light use of the channel with good separations between packets and, on the other hand, against team 6 that is one of the heavy users of the channel. The figures clearly illustrate the impact of playing against a heavy channel user team.



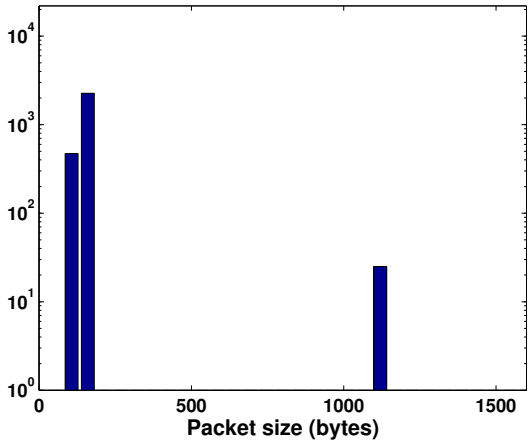
a) Team 1



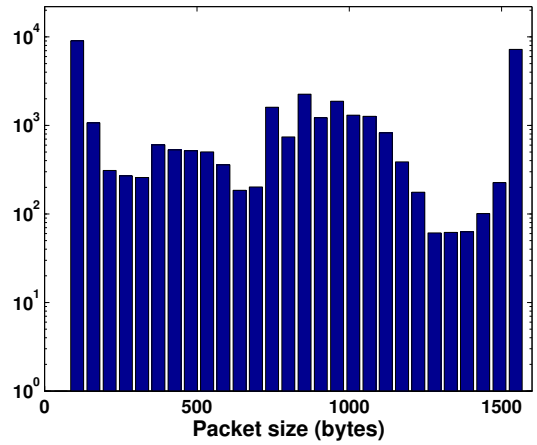
b) Team 2



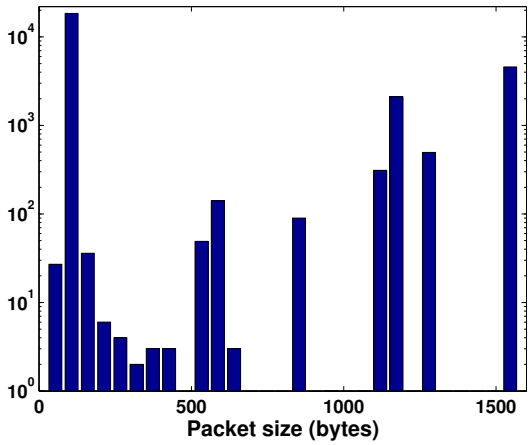
c) Team 3



d) Team 4



e) Team 5

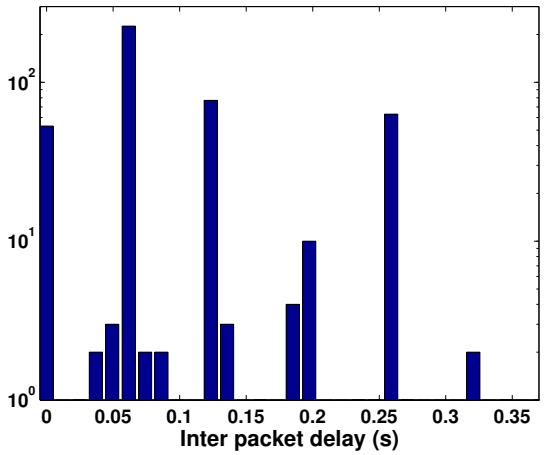


f) Team 6

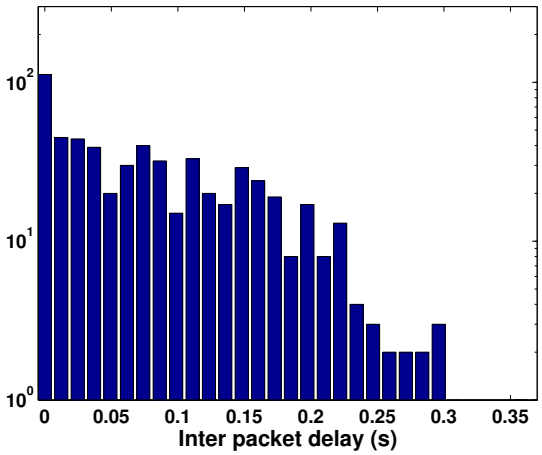
Figure 4.2: Histograms of packet sizes for each team

		Team 1	Team 2	Team 3	Team 4	Team 5	Team 6
Inter Packet	*avr*	17.74	15.20	20.03	21.72	1.74	1.90
(ms)	*std*	17.63	14.65	33.23	48.16	3.62	4.44
Packet Size	*avr*	412.87	139.68	160.51	187.67	787.40	497.81
(B)	*std*	73.66	8.03	5.59	93.77	549.09	598.36
Burst Size (# 1.5KB pk)		–	–	–	–	6	12
Total KB		1158	460	480	517	26154	13072
% of max		4.43	1.75	1.84	1.98	100.00	49.98
Bandwidth	802.11a	1.1%	0.4%	0.5%	0.6%	25%	13%
utilization	802.11b	5.5%	2.0%	2.5%	3.0%	125%	65%

Table 4.1: Traffic statistics of six MSL teams in RoboCup 2008



a) Robot 1 of team 2 against team 1



b) Robot 1 of team 2 against team 6

Figure 4.3: Inter-packet intervals for one robot of team 2 against teams 1 and 6

When playing against team 1, the traffic pattern shows a significant regularity, indicating negligible interference. However, the same robot playing against team 6 shows a significant change in the traffic pattern with a loss of the previous regularity and wide spread (strong jitter) of the inter-packet intervals, with a strong peak close to zero, meaning that many packets are strongly delayed and accumulated at the network interface, being then transmitted in burst.

Finally, it is important to refer that in these games the traffic external to the competition, including beacon frames from the AP, packets from other teams that were not playing, unknown packets, etc., was always negligible, representing less than 1% of the channel bandwidth. Another observation, during a different (non-monitorized) game than those monitored, was the use of raw (non-IP) packets by another team, which is also in violation with the rules.

4.3 Problems

In MSL, and probably in other RoboCup competitions as well, the wireless communication has always been a source of concerns, given the frequent occurrence of problems. These were of diverse kinds and could, in a simplified approach, be classified in four categories: infrastructure configuration, team communications configuration, lack of policing and channel overuse by teams.

- **Infrastructure configuration**. This category includes the cases in which the planning of the APs placement and channel assignment was non-optimal, frequently caused by constraints of the physical space in which the competitions must be layed out. Since it may be impossible to completely avoid this situation, it is necessary to live with a certain level of background interference, corresponding to an effective lower available channel bandwidth. Another problem is the interference with pre-installed WLANs for general Internet access, that must be switched off. This is a relevant issue that local organizers sometimes overlook.

- **Team communications configuration**. This has been one of the most common sources of problems due to frequent poor knowledge of the wireless communications technology. In fact, it is still common to find teams that bring their own APs and connect them freely in the team work area, often close to competition fields. Other times, the teams use erroneous configurations without being aware (e.g. ad-hoc mode), or send bursts of short packets overloading the network interfaces of the opponent team and causing some device drivers to crash, disabling communications and preventing a team from playing. Without an accurate analysis of the situation, the wrong team can be disqualified due to inability to play.

- **Lack of policing**. Despite being generally permissive, the MSL rules have dictated certain constraints for some time. Unfortunately, there was always a lack of policing to verify their effective application and enforcing them. In some years, the local organization has hired a specialized company to monitor and control the use of the

wireless channels. However, even in such cases it was difficult to mitigate all undesired situations, given their diversity, the number of wireless-enabled computers in the area, and the lack of rules compliance verification for the teams. The seek for spurious sources of interference might require the use of a specific wireless channel monitoring device that provides information on the channel status, not only at the network protocol level (transmitted valid packets) but also at the physical level (bit-error rate, spurious packet fragments, medium spectral analysis, ...).

- **Channel overuse by teams**. Even without spurious interference, when the channel utilization approaches high values the channel performance deteriorates in terms of packet transmission delays and packet losses due to increased collisions and channel saturation. These delays and losses have a direct negative impact on the quality of the cooperating behaviors given their real-time character, mainly when they involve feedback control over the wireless channel. Transient saturation must also be considered and prevented, such as caused by bursts transmitted by the same station, e.g., raw image transfers. These can also cause a transient increase in packet delays and losses suffered by the opposing team that can harm the performance of its cooperative applications. To prevent these situations the teams must adhere to some kind of control of the consecutive amount of data that each of their robots transmits in an agreed interval of time. On the other hand, detecting such situations requires monitoring the traffic with increased temporal resolution.

4.4 Common misconceptions

As it was clear with the previous discussion, most of the problems can be solved or strongly attenuated with adequate restrictions on the use of the wireless channel and an effective policing of the channel utilization. Nevertheless, it is interesting to quickly analyze certain misconceptions that hindered the deployment of such solutions:

- **No need for restricting teams transmissions**. Ideally, if the channel bandwidth was infinite and there was no mutual interference between the competing teams, restricting the teams transmissions would make no sense. However, that is not the case, and finite bandwidth and mutual interference are facts that need to be considered. As shown in the section 4.2, while some teams do a parsimonious use of the channel, others exist that use substantial amount of bandwidth, often in a bursty way, with negative impact on the timeliness of the transmissions of the opposing team and consequently on the performance of its cooperative behaviors. Thus, some form of restriction that

considers both bandwidth and bursts must be enforced. For example, teams must adhere to some kind of control of the consecutive amount of data that each of their robots transmits in an agreed interval of time.

- **Larger bandwidth solves the problem**. Unfortunately, just increasing the available bandwidth alone, as when moving from IEEE 802.11b ($11 Mbps$) to IEEE 802.11a ($54 Mbps$), is not a self-sustained solution and tends to generate wasteful patterns in bandwidth utilization. Such kind of simplistic solutions is always transitory and end up coming back to the same problem but with a larger magnitude. This trend was verified with teams 5 and 6 as shown in the previous section.

- **Use a technology with QoS support**. In order to provide better support to time-sensitive traffic with respect to non-time-sensitive one in WLANs, IEEE 802.11e was proposed. Similarly to the original protocol, it includes two channel access policies, one that is distributed - Enhanced Distributed Channel Access (EDCA) - and another one that is controlled - Hybrid Coordination Function Controlled Channel Access (HCCA). The former is the one that is starting to be accessible commercially while the latter has not received significant adherence by equipment manufacturers. Unfortunately, the latter is also the one that could bring more advantages to the RoboCup environment since it allows creating isolated channels with negotiated bandwidth, thus without mutual interference. The former just creates prioritized traffic classes, which does not help since, within a game, one team cannot be prioritized with respect to the other and rules would still be needed to guarantee fairness when sharing the same priority class. Moreover, there would be no guarantee that other external sources of interference would not transmit at the same or higher priority level, thus not avoiding the interference problem. Since it is not clear whether equipment supporting HCCA will ever be available due to market reasons, and its expected higher cost, it seems unnecessary to change the current technology. Instead, it seems worth working on enforcing appropriate bandwidth sharing policies and mechanisms.

- **No need for technical verifications**. Ideally, teams should verify and enforce compliance of their equipment with the rules. However, in some cases, particularly with the wireless communication technology due to its idiosyncrasies, the teams often lack the knowledge to adequately enforce the needed configurations. Without technical verifications before the actual competitions, those problems will be discovered in the game, only, and will be hard to diagnose correctly.

Finally, the middleware used also has a significant impact. For example, using multicasts in a producer-consumer style allows a faster dissemination of the information, with better synchronization, for four or more stations, on average. A preliminary study of the effect of using multicast/broadcast packets versus unicast ones in a multi-robot scenario is shown in [85]. Direct pair-wise exchange of information, in a peer-to-peer fashion, tends to generate much more traffic for disseminating the same information. Similarly, the use of a central Blackboard used in a client-server fashion requires about twice the transmissions than a corresponding producer-consumer model with direct robot-to-robot communication.

4.5 Summary

In this chapter we presented an analysis of the issues related to the wireless communications in the RoboCup MSL. This analysis was based on measurements in a real competition. While some problems are related to poor layout and/or configuration, other arise from bad usage by teams, typically associated with channel abuse.

Therefore, we believe that following appropriate programming practices will have a strong positive impact on the quality of the communications. The main best practices that we suggest are:

- **Middleware that minimizes transmissions** – Among the diverse types of middleware available, as discussed in Chapter 3, using one that relies on multicast/broadcast transmission, such as a Publisher-Consumer (PC) or a PS model, will reduce the network utilization and thus flavor a more efficient use of the channel;
- **Robot and team control based on state** – Opting for a control based on states, instead of events, typically generates periodic or quasi-periodic traffic patterns that are amenable to coordination and better packing under high channel utilization. On the other hand, events tend to generate bursts of communication that lead to moments of high mutual interference;
- **Low bandwidth cooperation** – Using cooperation approaches that rely on minimal information exchange is not only more robust with respect to network problems but also contributes to a healthy network with low to medium utilization. Note that the lower the network utilization the shorter will be the incurred delays and packet losses.

Using these best practices as requirements to an adequate communications solution for RoboCup MSL, we propose a lightweight middleware for state based control with periodic data exchange. Moreover the communication protocol is able to adapt to instantaneous

interference by shifting the phase of its periodic communications, and it is also capable of automatic reconfiguration following the team current composition. In this protocol, the team communicates in a cycle that is essentially constant, except for a short tolerance to cope with external interference. The cycle is then divided at each moment according to the actual number of robots and their communications are always separated as much as possible and equally divided by all the active team members, creating periodic time gaps for the opponent team to communicate. This proposed solution is the core of our thesis and will be explained in detail in the remainder of the dissertation. We believe that this middleware and wireless communications protocol are a general solution to all situations in which a robotic team entails strong interaction among its members under potential external interference in an area of few tens of meters wide.

Chapter 5

The Reconfigurable and Adaptive TDMA communication protocol

Using wireless communication to support cooperation among members of a team of robots raises several challenges. In fact, the wireless medium is open and prone to errors and it is also fast fading leading to limited communication range and other problems such as hidden nodes and exposed nodes. Moreover, being an inherently shared medium, some kind of access control is needed.

The option for an existing wireless communication standard solves those problems up to a certain extent. In the Robotics community the most common protocol is IEEE 802.11 in infrastrutured mode[20] and this is also the base technology that we will use. In the infrastrutured mode all the communications pass through an AP, as described in Section 2.1. This brings some benefits in terms of team membership consistency, which is enforced by the AP. An agent is considered as part of the team when it has an active link with the AP, not necessarily a link with each of the other agents. The coverage range of the team can be extended placing the AP in the middle of the operational field. This is a realistic option in many application scenarios. For example, for teams of surveilling robots within large indoor spaces, such as malls, it is normally feasible to provide an AP that guarantees the radio coverage of all robots. In de-mining applications, or even search and rescue, it is possible to place the AP on top of one of the robots deployed near the center of the operations area that will provide coverage for the other ones.

[20]Despite the prevalence of the infrastrutured mode, there are many applications that use IEEE 802.11 in ad-hoc mode such as mining robots, or search and rescue robots, in which it is relevant to allow long topologies without compromising connectivity.

Nevertheless, there are two general concerns with respect to using the wireless channel. On one hand, it is always desirable to reduce transmissions to the minimum possible, which has a positive effect on keeping the communication load low leading to a better network behavior in terms of packet delivery and latency. On the other hand, despite the existence of distributed arbitration mechanisms in IEEE 802.11, access collisions can still occur and their probability raises significantly with the network load.

This chapter presents a communication protocol that organizes the communications of a team of nodes in an IEEE 802.11 infrastrutured network so that the probability of collision is reduced. In particular, the team communications are automatically separated as much as possible in a periodic TDMA framework. However, as opposed to traditional TDMA schemes, the protocol does not rely on clock synchronization and allows adjusting to variations in the team composition.

The resulting periodic traffic pattern is rather permeable, with maximized intervals between team transmissions, which also alleviates congestion at the network access and makes the protocol resilient to external traffic, i.e., uncontrolled traffic sent by IEEE 802.11 stations outside the team.

Moreover, the protocol also includes a phase adaptation scheme that allows escaping from coherent periodic interfering sources. These sources, which have periods that are integer multiples or submultiples of the protocol period, can have a significant negative impact in network performance even with low load.

5.1 TDMA communications

TDMA is a common temporal multiplexing scheme for periodic communications in which each node has a dedicated fixed duration transmission window or slot. These slots are then organized in a round that repeats continually in time. Since transmissions from different nodes are separated in time (they occur in different slots) the occurrence of collisions when accessing the medium is precluded.

Given the periodic round pattern, the typical implementation of TDMA schemes relies on a clock synchronization service that allows determining and enforcing the occurrence instants of all slots into the future as well as synchronizing other activities in the nodes with the communications schedule. This global synchronization is typically referred to as a Time-Triggered Architecture [48] and can be advantageous for reducing end-to-end latencies

of distributed behaviors. The down side is a significant complication of the global system configuration since triggering instants must be defined for all activities in all nodes and network, which is not always possible.

In order to tolerate uncontrolled external traffic two requirements must be fulfilled. On one hand, there must be a collision resolution mechanism since access collisions are now possible. This is granted by the underlying IEEE 802.11 protocol. On the other hand, the nodes in the team must use windows that are sufficiently larger than their own transmissions to make space for the external traffic (Figure 5.1). Under heavy external traffic load, it is still possible that the transmissions of one node in the team fall inside a following window, creating interference within the team.

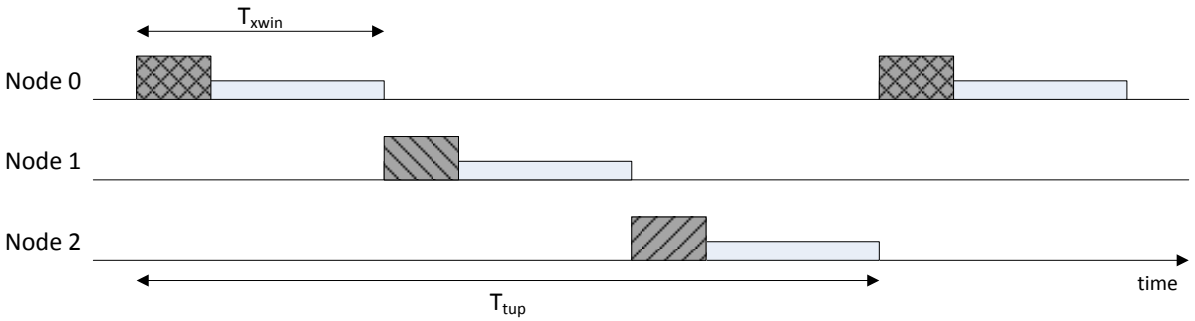


Figure 5.1: TDMA round

Figure 5.2 shows a practical case with four robots and a period of $99.5ms$, in which clock synchronization was achieved with Chrony[21]. In particular, the top part of the figure shows the offset of the transmissions of each node in the team as observed by an arbitrary reference node. The top line shows the offset of the following transmission of the reference node itself, thus the actual round duration as observed from the communications. In this experiment there is a residual load of uncontrolled external background traffic, plus a periodic interference caused by an external node issuing a `ping` command to the AP with $1KB$ every $5ms$. The interferences in the offsets are clearly visible in the spikes that affect the respective lines.

The lower part of Figure 5.2 shows the instantaneous network load. It is visible that there are approximately periodic spikes caused by relative drifts between the team clock and the clock in the node generating the interference. The spikes occur when the `ping` packets are sent very close to the team packets. On the right we can see a histogram of successful team transmissions, single lost packets and multiple consecutive lost packets.

[21] `http://chrony.tuxfamily.org`

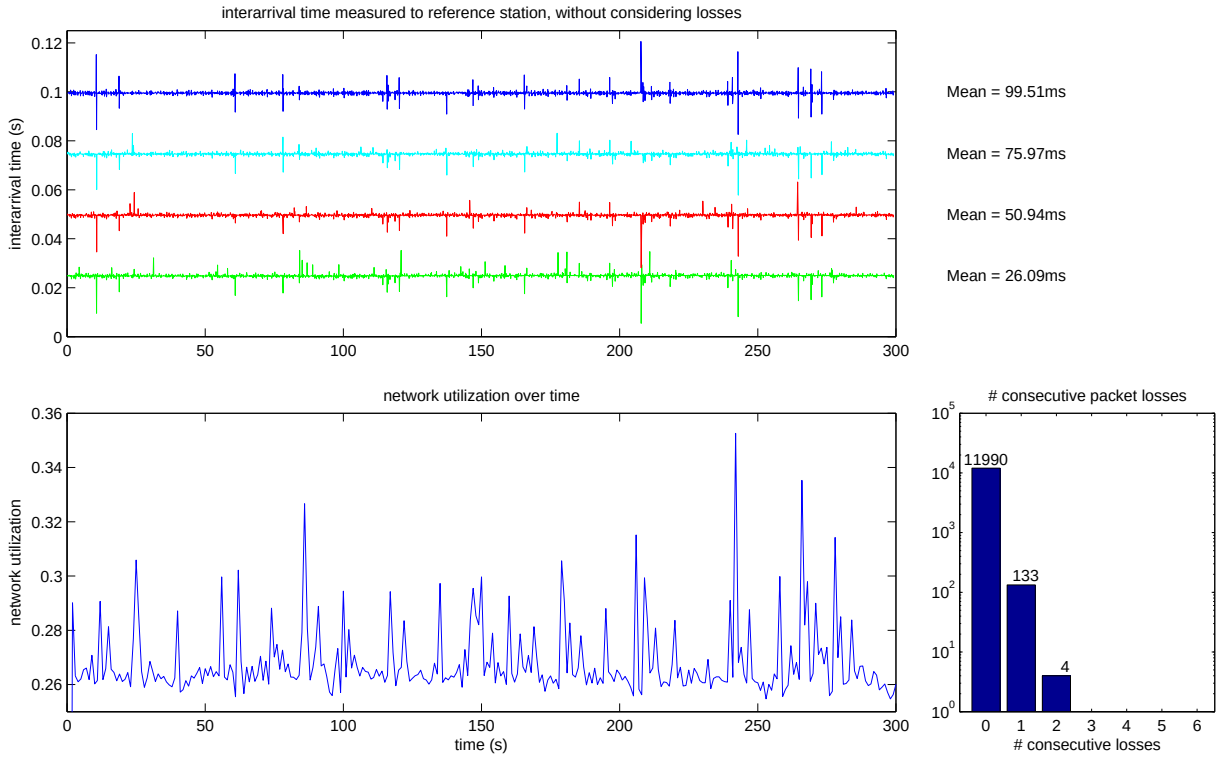


Figure 5.2: TDMA with round of 4 robots, 100ms period and periodic external interference. Top: slots offsets in the round; Bottom left: instantaneous network utilization; Right: histogram of consecutive packet losses

The pernicious situation caused by periodic interference is shown in Figure 5.3. Given the fixed TDMA round structure, a periodic interference with a coherent period, will cause persistent interference, independently of the load, increasing the probability of collisions and consequent packet losses.

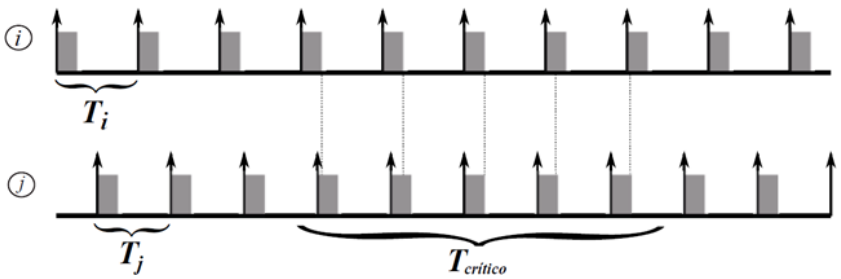


Figure 5.3: Interference of a coherent periodic source in a TDMA framework [78]

In fact, this pernicious phenomenon also happens when the robots in the team transmit periodically, and with similar periods, but unsynchronized. In this case, each one will be a coherent periodic interference for the others. Experimental results presented later on show the negative impact of this phenomenon in terms of packet losses and network latency, establishing the value of synchronization in these scenarios.

5.1.1 Configuring the TDMA framework

As mentioned before, one concern that must be taken into account in the context defined above is the reduction of the communication load generated by the team. This led to two options, namely a aggregation of the information to send in as few packets as possible, and the use of multicast transmissions.

Packetization of the information to send

One way to reduce the team generated network load is to have each node aggregation the information that it wishes to send in a small number of packets, thus saving overhead. In fact, in all operational scenarios used thus far, one single packet has been enough.

However, if the packets are too large, say above half the typical Maximum Transmission Unit (MTU) of $1500B$[22], the probability of packet loss also increases significantly and the savings in overhead do not compensate. This however, depends on the specific error rates encountered in the operational scenario.

The protocol, as we have descrived it thus far, does not include fragmentation and reassembly of large amounts of information. However, using this protocol under a UDP/IP network stack solves this problem, since it already includes fragmentation and reassembly of up to $64KB$. Nevertheless, care must be taken to make sure the packets fit inside the respective node slot and still leave sufficient free bandwidth for any expected external traffic[23].

Transmissions in multicast mode

On the other hand, the team generated communication load can also be reduced using multicast transmissions. In this case, the transmissions from each individual node up to the AP are still unicast, taking advantage of the increased reliability of possible retransmissions. However, the transmissions down from the AP are in multicast for all team members at once.

The use of multicast transmissions has the advantage of enforcing synchronization of the multiple receivers upon packet reception and transferring the respective information to multiple receivers with just one network transaction, making it scalable. On the other hand, multicast transmissions are unacknowledged, as opposed to unicast ones, thus being less reliable. Moreover, they are typically transmitted at a lower rate, thus using more bandwidth.

[22] Note that the MTU, or payload, for Wi-Fi is $2312B$ but it is typically set to $1500B$ for consistency with Ethernet.

[23] A generic interface that transparently confines any amount of traffic to the node assigned slot is under way, see future work.

However, depending on the selected transmission rate for the multicast traffic, this mode already compensates. Figure 5.4 shows the possible multicast transmission rates and the minimum number of nodes in the team to compensate bandwidth wise, with respect to a logical multicast achieved with multiple unicast transmissions.

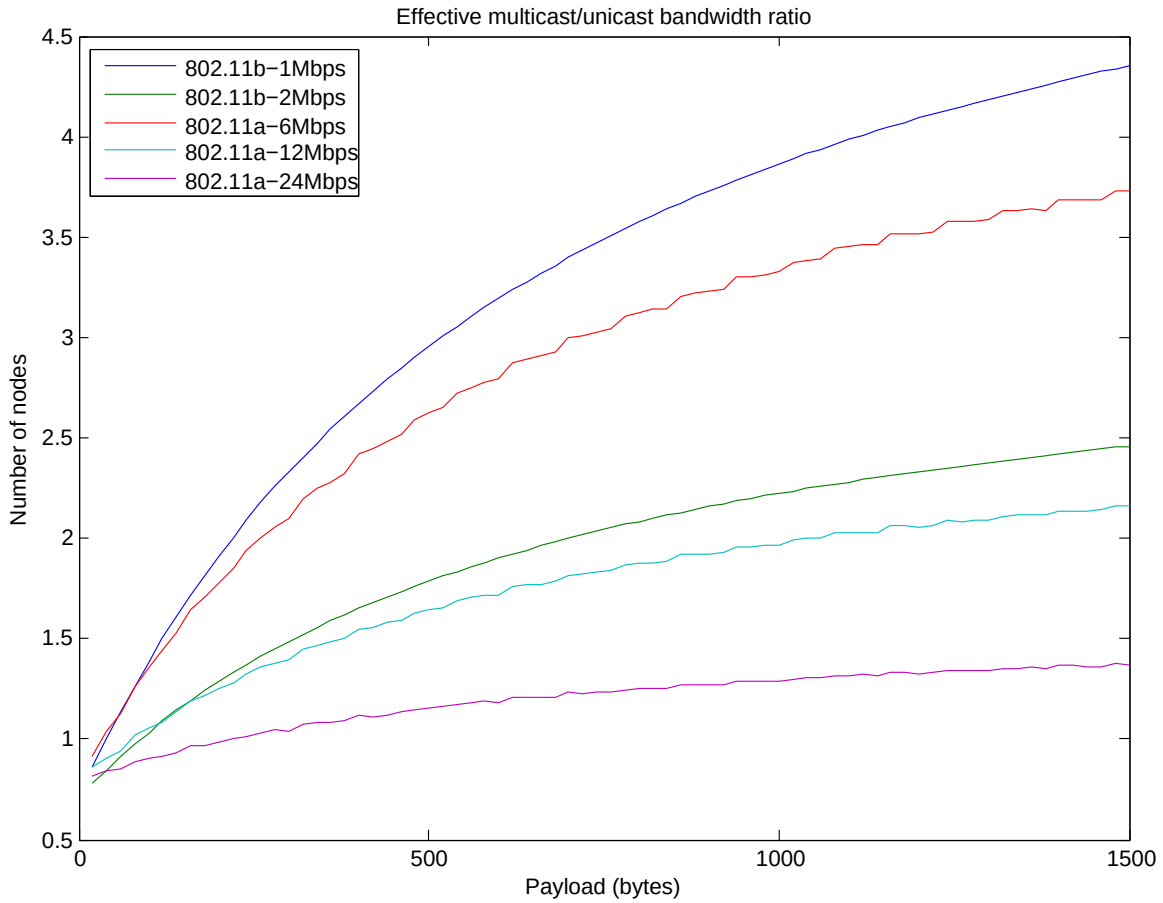


Figure 5.4: IEEE 802.11 effective multicast/unicast bandwidth ratio

In this respect, it is important to note that lower transmission rates typically improve transmission reliability but also increase packet transmission latency. The best compromise depends on the typical error rates of the actual operational scenario. We have successfully used either $6Mbps$ or $24Mbps$[24].

[24] An example of an empirical study in a mine scenario applied to IEEE 802.11a in ad-hoc mode [89] has shown that $24Mbps$ is a good option in that case.

Setting the TDMA period

The TDMA period is typically a configuration parameter, set offline. In our context, in which the transmissions are used to share the robots' internal state information with the other members of the team, we call it the *team update period (T_{tup})*. This parameter has an important impact in the real-time performance of the network, setting the temporal resolution of the global communication and affecting the responsiveness of the team.

With respect to the responsiveness, as in any polling system, any change occurred in the internal state of a node immediately after a state transmission will need to wait for the next slot, i.e., approximately one period, to be transmitted. For this reason, when there is a sporadic event that requires urgent transmission, we send it immediately as traffic outside the protocol. For seldom urgent transmissions, this has practically no impact on the TDMA framework.

On the other hand, the regular state updates associated to the recurrent sensing and control activities are disseminated at most once each round, which, thus, sets the temporal resolution of those updates.

As is typical in regular TDMA frameworks, the round is divided equally by the number of team members leading to the TDMA slot structure. This sets a relationship between the round duration, T_{tup}, the number of robots, N, and the width of the slots, T_{xwin}. This is expressed in Eq. 5.1.

$$T_{xwin} = \frac{T_{tup}}{N} \tag{5.1}$$

Being an important parameter for cooperative behaviors, T_{tup} must be set considering the respective real-time requirements. In general, T_{tup} should take the maximum value that allows meeting those requirements. Maximizing T_{tup}, for any given number of nodes, also maximizes T_{xwin} and thus the tolerance to external traffic.

If we know the expected load imposed by the external traffic, and our communication requirements, then we can also determine the minimum T_{tup} that allows accommodating both. Eq. 5.2, equivalent to 5.3, gives a simplified lower bound for T_{tup} where D_i is the maximum number of data bytes that node i transmits in any round, $frameType$ can be u for unicast, m for multicast or b for broadcast, $netType$ is the IEEE 802.11 profile, namely a, b or g. Then, the function $nodeLoadTime()$, described in Annex A, returns the time taken by the transmission of D_i bytes by node i in each round and the function $extLoadOcup()$, also described in Annex A, returns the fraction of network occupation taken by the external load L expressed as a required throughput ($Mbps$).

$$T_{tup} > \sum_{i=0}^{N-1} \Big(nodeLoadTime(D_i, frameType, netType) \Big) + \\ + extLoadOccup(L, frameType, netType) * T_{tup} \tag{5.2}$$

$$T_{tup} > \frac{\sum_{i=0}^{N-1} nodeLoadTime(D_i, frameType, netType)}{1 - extLoadOccup(L, frameType, netType)} \tag{5.3}$$

In practice, this expression does not account for retransmissions, but it already considers an average transmission rate lower than the maximum as well as an average backoff delay equal to half of the initial contention window. Nevertheless, even if this expression was accurate, the associated boundary condition would lead to a nearly saturated network, which results in poor behavior due to collisions and overload by retransmissions.

Thus, it seems more reasonable to compute a value for T_{tup} that leads to a target maximum network utilization of $\Omega < 1$, which can be achieved using Eq. 5.4,

$$T_{tup} > \frac{\sum_{i=0}^{N-1} nodeLoadTime(D_i, frameType, netType)}{\Omega - extLoadOcup(L, frameType, netType)} \tag{5.4}$$

For example, consider the case of the CAMBADA RoboCup MSL team in 2008 (when the logs in Chapter 4 were taken), transmiting in IEEE 802.11a multicast, with 6 robots and generating $5 \times 354B + 548B$ of data per round. This represents approximately $6ms$ of traffic according to the function $nodeLoadTime()$. Now, consider a game against team 6, a team that aggressively uses 13% of the network capacity. If we vary the target total network load Ω from 1 down to 0.25, the resulting values for the lower bound on T_{tup} are shown in Fig. 5.5, going from $6.9ms$ to $50ms$. If the actual real-time constraints of the cooperative behaviors allow raising T_{tup} to $100ms$, the resulting total network load will be approximately 19%. This value may already cause significant perturbations to a team traffic, as reported in Fig. 4.3 of Chapter 4. As mentioned, team 6 issues an interference load of 13% of the network capacity.

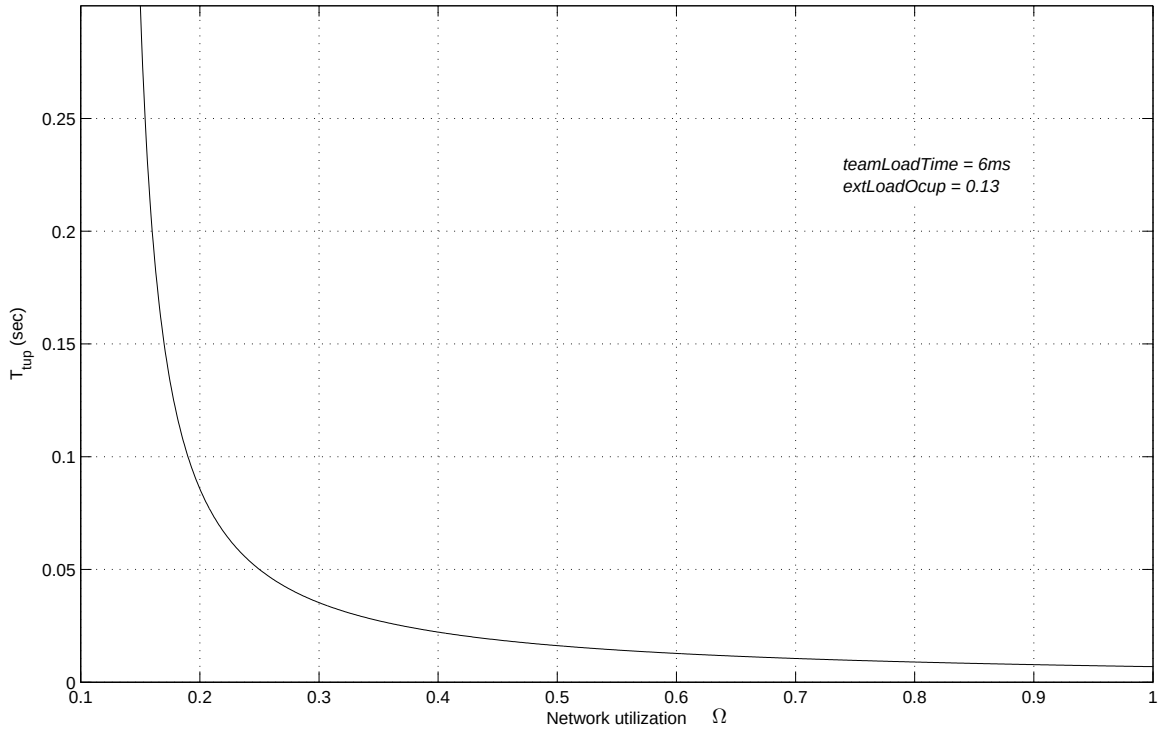


Figure 5.5: Lower bound on T_{tup} as a function of the total network load Ω

5.2 Adaptive TDMA

One of the problems of the ordinary TDMA framework presented before is the susceptibility to situations of periodic interference like those illustrated in Fig. 5.3. In fact, given the absence of any adaptation mechanism, the protocol will continue generating transmissions at the same time as the interference source for a while, until the clock drifts eventually set those instants apart.

This was the motivation to develop the Adaptive TDMA protocol which is sensitive to the delays suffered by team members and uses such delays to rotate the phase of the TDMA round. In a situation of periodic interference, a transmission of a team member would eventually be delayed by the interfering source. The remaining members of the team would detect such delay and would then delay their own transmissions by the same amount of time, effectively shifting the phase of the TDMA round so that the following transmissions would not collide with the interfering source again.

Moreover, this technique also carries along a further benefit since each node now synchronizes with the others in the team by measuring the reception instants of the respective packets and thus clock synchronization is no longer needed.

Formally, when the timer that triggers the transmissions in node j fires at time $t_{j,now}$, it issues its transmission and sets the timer to fire at $t_{j,next} = t_{j,now} + T_{tup}$, i.e. one round after. However, during this interval, it continues monitoring the arrival of the packets from the other nodes in the team. When the packet from node i arrives, the delay δ_i of the effective reception instant with respect to the expected instant is determined. If this delay is within a validity window $[0, \Delta]$, with Δ being a global configuration parameter, the next transmission instant is delayed according to the longest such delay among all the packets received from the team in one round (Figure 5.6), i.e.,

$$t_{j,next} = t_{j,now} + T_{tup} + max(\delta_i)_{i=0..N-1, i \neq j \wedge \delta_i \leq \Delta} \tag{5.5}$$

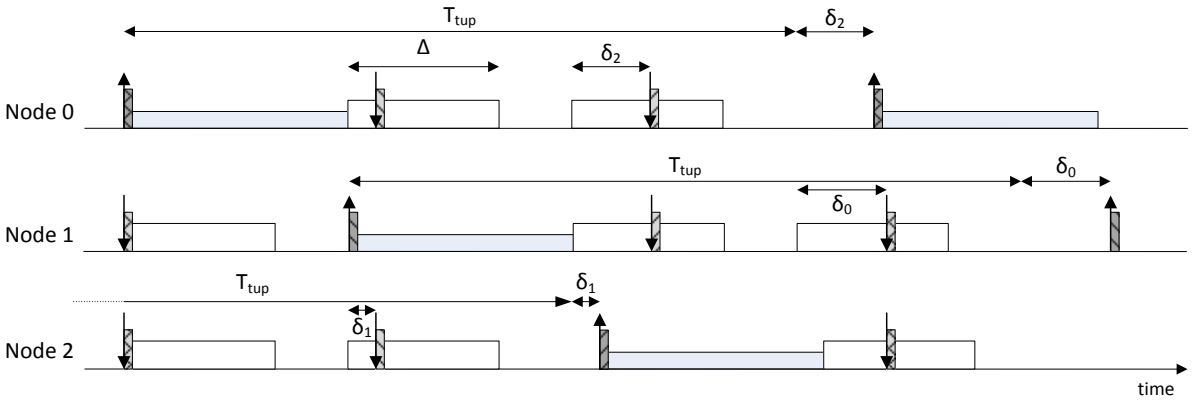


Figure 5.6: Adaptive TDMA round

On the other hand, if the reception instant is outside that validity window, then δ_i is set to 0 and does not contribute to update $t_{j,next}$.

The practical effect of the adaptation in the protocol is that the transmission instant of a packet in each round may be delayed up to Δ with respect to the predefined period T_{tup}. Therefore, the effective period will vary within $[T_{tup}, T_{tup} + \Delta]$. From a real-time requirements perspective, the constraints on the round update should now be applied to $T_{tup} + \Delta$.

With this method the protocol makes run-time adaptations to the current network load, increasing the effective update period when there are significant delays affecting the team transmissions, which corresponds to a small reduction in the team generated load, contributing to stabilize the network.

Figure 5.7 shows a practical case with four robots and a period of $99.5ms$, equivalent to the one shown in Figure 5.2 but using Adaptive TDMA. The top diagram shows the offset of the reception instants of the transmissions of each node in the team also with respect to

an arbitrary reference node. The top line shows the offset of the following transmission of the node used as reference. This situation includes, beyond a residual load of uncontrolled external background traffic, a periodic interference caused by an external node issuing a `ping` command to the AP with $1KB$ every $5ms$. The interferences in the offsets are clearly visible but, in this case, the offsets tend to be above the configured value due to the increments in the period made by Adaptive TDMA.

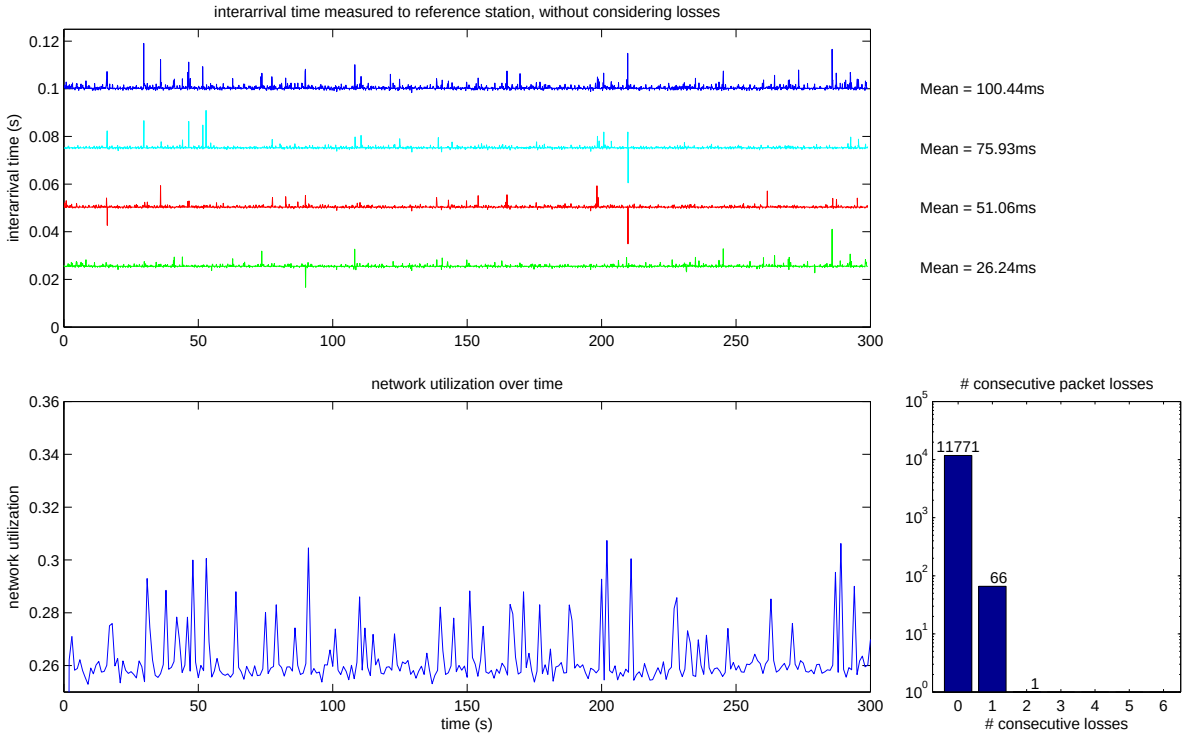


Figure 5.7: Adaptive TDMA with round of 4 robots, $100ms$ period and periodic external interference.
Top: slots offsets in the round; Bottom left: instantaneous network utilization;
Right: histogram of consecutive packet losses

The bottom diagram of Figure 5.7 shows the instantaneous network load. The periodic spikes visible in Figure 5.2 do not occur here since the protocol quickly moves away from the interference source, thus avoiding any periodic interference pattern. On the right we can see a histogram of successful team transmissions, single lost packets and multiple consecutive lost packets. Here, we can observe a significant reduction in the number of lost packets. We conjecture that this reduction is due to the adaptation mechanism of Adaptive TDMA. More results will be shown further on, in Chapter 7.

5.2.1 Additional protocol configurations

The Adaptive TDMA protocol inherits the same configurations described for the ordinary TDMA framework, e.g., in the definition of T_{tup} and use of multicast packets. However, it is extended with other parameters and features related with the adaptation capability.

Setting the validity window

The main parameter specific to the adaptation of the protocol is Δ, the validity window, that filters the delays suffered by the transmissions of the team members. Δ is predefined offline, similarly to T_{tup}, and it defines the stretchability of the protocol. For reasons that will become clear further on, the value of Δ is not defined in absolute terms but as a fraction of the transmission window (T_{xwin}) in the TDMA round (Eq. 5.6).

$$\Delta = T_{xwin} \times \epsilon \ , \ 0 < \epsilon < 1 \tag{5.6}$$

Consequently, this relative definition creates a new parameter ϵ that sets the actual value of Δ. An ϵ closer to 1 creates a Δ closer to T_{xwin} allowing the protocol to adjust more to the network load but, conversely, it allows larger variations in the effective period with a possibly noticeable reduction in the system responsiveness. On the other hand, an ϵ closer to 0 leads to a smaller Δ that reduces the protocol capacity to adjust and, ultimately, can prevent the protocol from synchronizing the team nodes in the common TDMA round.

In our experiments we have frequently used an empiric value of 66% but it must be adjusted to each case, essentially depending on the magnitude of the delays suffered by the team transmissions. Higher external load imposes larger delays requiring a larger ϵ to increase the probability of keeping the team synchronized.

Transmitting multiple packets

With the Adaptive TDMA protocol it is also possible to send multiple packets in each slot. As pointed out before, it is necessary to ensure that such transmissions are confined to the slot, to avoid interference within the team. However, within Adaptive TDMA there is another concern since the packets are also used for synchronization purposes.

The way we use to allow multiple packet transmissions is simple. The first packet transmitted in the slot is marked so that it is used for synchronization purposes. However, any following packets in the same slot are considered just for the data they carry but ignored for synchronization purposes.

Fully distributed node resynchronization

When node j does not receive any packet in a round within the respective validity windows, it updates $t_{j,next}$ using a node specific configuration parameter β_j as described in Eq. 5.7.

$$t_{j,next} = t_{j,now} + T_{tup} + \Delta + \beta_j \ : \ 0 < \beta_j < \Delta, \beta_j \neq \beta_i \ : \ j \neq i \tag{5.7}$$

This specific parameter is used to force different effective transmission periods, generating a sliding phase relative to the other team members thus preventing a possible situation in which the nodes would all remain transmitting but unsynchronized, i.e., outside the validity windows of each other, and with the same period T_{tup}. By imposing different periods, the nodes are forced to re-synchronize within a limited number of rounds because the transmissions will eventually fall within the validity windows of each other.

We call this phase rotation the *scan mode* during which an unsynchronized node uses a slightly longer update period to rotate its relative phase in the round, see Figure 5.8. The parameter β sets the speed of phase rotation and since it is different among all the nodes, the amount of time that each node needs to find its own transmission window is also distinct. The number of rounds to synchronize should vary uniformly between 0, when the node, by chance, starts scanning exactly in the slot that was assigned to it, and a certain upper bound, R, that depends directly on β as expressed in Eq. 5.8.

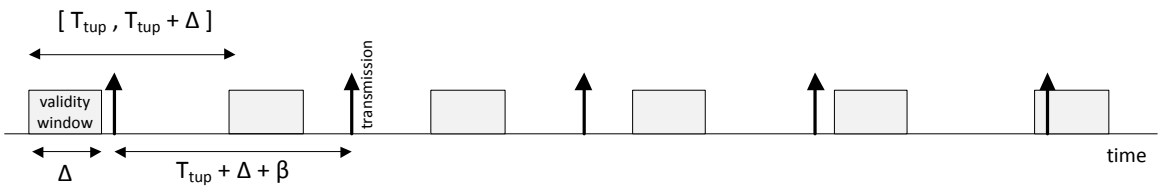


Figure 5.8: Phase rotation to achieve synchronization

$$R \leq \left\lceil \frac{T_{tup}}{\beta} \right\rceil \tag{5.8}$$

Note that this expression is rather pessimistic since it assumes that the Adaptive TDMA mechanism is constantly rotating with the maximum period of $T_{tup} + \Delta$, which would only occur under heavy external load. However, when the external load is low, the actual TDMA round will be close to T_{tup} and thus the maximum R will be closer to $\frac{T_{tup}-\Delta}{\beta}$.

Figure 5.9 shows a log from an actual run in which only two nodes were active in a team of ten. This log shows the offsets of the nodes transmissions with respect to an arbitrary reference, i.e., one of the two. The diagonal parts of the traces show the scan mode occurring upon loss of synchronization caused by a sufficiently strong delay.

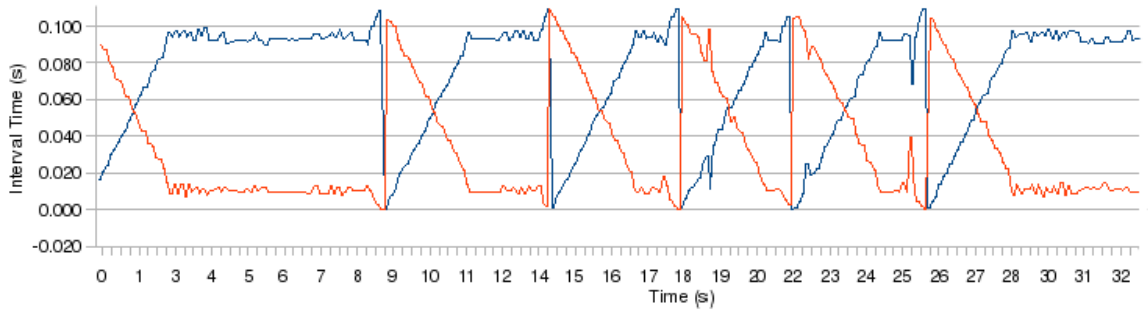


Figure 5.9: The scan mode in an actual run with two nodes in a ten slots TDMA

5.2.2 Limitations of the fully distributed resynchronization approach

The fully distributed approach to resynchronize the nodes upon an asynchronous restart, or loss of synchronization was in place and used for some time in the CAMBADA MSL team. However, it eventually revealed some limitations. As presented previously, each time a node fails its validity window, it starts transmitting unsynchonized, i.e., outside of its transmission window, and takes up to R rounds to reach synchronization, as given by Equation 5.8.

A worst case situation was discovered during strong network interference, revealing the occurrence of cliques. When more than one agent is transmitting unsynchronized at the same time, thus in the *scan mode*, it is possible that one would synchronize with the other and, from then on, both would consider to be synchronized and exit the scan mode, despite being unsynchronized with the rest of the team. This occurrence of cliques is undesirable since the transmissions of nodes in different cliques could cause interference to each other, thus interference within the team.

Another limitation arises from the use of a fixed number of team members N, even when several are inactive at a certain point in time. This leads to T_{xwin} values smaller than needed, i.e., the slots are unnecessarily short since some of them are not used. Note that a smaller

T_{xwin} reduces the leeway to accommodate delays caused by the external traffic and thus increases the probability of loss of synchronization and possibly leading to collisions within the team.

Another problem discovered during tournaments was the interference with the power management mechanisms of the APs as explained in Section 2.1. With these mechanisms, multicast packets are buffered in the AP and released every DTIM interval right after the beacon transmission. This imposes rather large delays in the reception of the nodes multicast packets that only by chance fall within the respective validity windows. Consequently, the protocol will fail to synchronize.

Interestingly, some APs apparently do not allow switching off the power management mode, which was the case with those used in some of the RoboCup competitions, thus raising our awareness of the problem.

The solution to these limitations is discussed next.

5.2.3 Resynchronizing with a fixed reference

The shortcomings arising from the resynchronization approach presented previously were solved by using only one agent as reference, thus moving from a fully distributed to a centralized approach. With this method, all the nodes synchronize to the same reference node, using the reception instants of its packets, and only this node is in charge of adapting the TDMA round phase to the current network load (Figure 5.10).

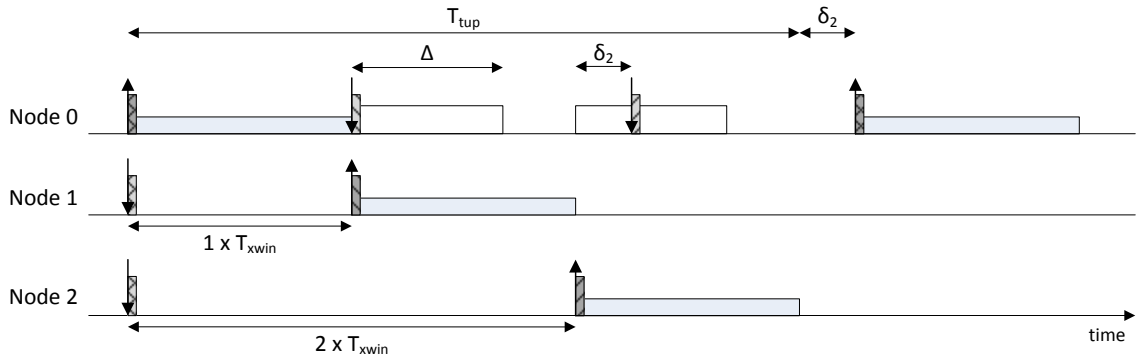


Figure 5.10: Enhanced Adaptive TDMA round using agent 0 as reference

This approach prevents the formation of cliques because the synchronization reference is unique. Moreover, since the transmissions instants are defined with offsets to a common reference, they continue being effectively separated in their slots avoiding mutual interference within the team. This is the case even under interference of the DTIM power management

mechanism. The impact of DTIM is just an enlargement of the effective period caused by the possible extra delay affecting the reception of the reference packet. This can be easily determined and accounted for by consulting the AP configuration.

In this case, all nodes compute their next transmission instant based on the reception of the packet from the node used as reference and an appropriate offset to the respective slot (Eq. 5.9).

$$t_{i,next} = t_0 + i \times T_{xwin} \qquad i \neq 0 \tag{5.9}$$

In this case, node 0, used as reference, determines its next transmission using a slightly modified version of Equation 5.5 as expressed in Equation 5.10.

$$t_{0,next} = t_{0,now} + T_{tup} + max(\delta_i)_{i=1..N-1,\delta_i \leq \Delta} \tag{5.10}$$

With this new resynchronization approach, the β parameter used previously to create phase rotation is not necessary and any node can find its slots immediately after receiving a packet from the reference node. Thus, the previously mentioned scan mode no longer exists and resynchronizations are faster, taking at most one round.

5.3 Dynamic reconfiguration of the TDMA round

As stated before the use of a fixed number of nodes, N, independently of whether they are available or not, leads to a smaller T_{xwin} when compared to the optimal value based on the total number of actually running agents. It also creates difficulties in synchronization and, for nodes transmitting in contiguous slots in the TDMA round, the probability of mutual interference is higher.

5.3.1 Recomputing parameters based on the actual number of nodes

Therefore, we added a self-configuration capability to the protocol, to cope with variable number of team members and adapt the TDMA round accordingly. This specific mechanism supports the dynamic insertion/removal of nodes in the round in a fully distributed way.

The T_{tup} period continues to be constant, as determined by the real-time constraints of the cooperative behaviors. However, it is divided equally among the running nodes at each instant, designated K, with $K \leq N$, maximizing the slot width, T_{xwin}, at each moment, thus leading to maximal separation between the transmissions of the nodes in the team. Equation 5.1 is then re-written as in Eq. 5.11.

$$T_{xwin_K} = \frac{T_{tup}}{K} \tag{5.11}$$

The validity window used in the TDMA adaptation also becomes dynamic and a function of K, namely Δ_K as given by Eq. 5.12. Note that its actual value follows the variations in the width of the slot according to the fraction defined in the configuration parameter ϵ.

$$\Delta_K = T_{xwin_K} \times \epsilon \ , \ 0 < \epsilon < 1 \tag{5.12}$$

However, the number of active team members K is a global variable that must be consistent so that the TDMA round is divided in the same number of slots in all nodes. To enforce consistency in the adaptation of the current number of active team members a membership vector was added to the packet transmitted by each node in each round, containing its perception of the team status (see Figure 5.11). The number of fields in the membership vector is the maximum number of team mates N, defined off-line.

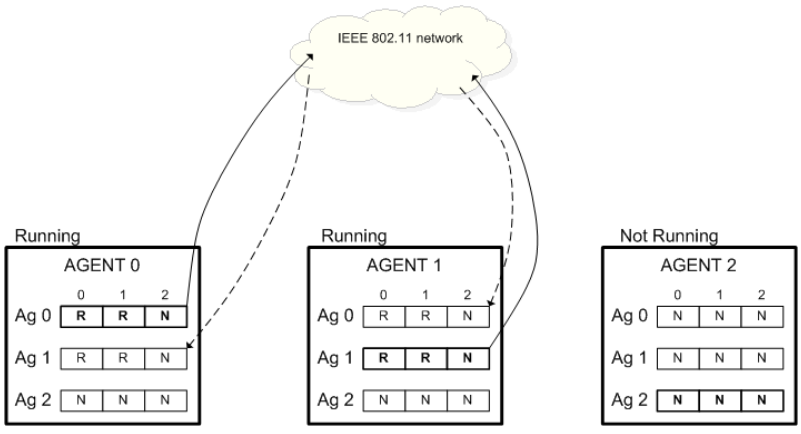


Figure 5.11: Dissemination of the membership vectors

One important aspect concerns the identification of slots and nodes. In the Adaptive TDMA mechanism, each node had a unique ID ($0 \leq i \leq N-1$) that was also used as slot ID. However, with the reconfiguration mechanism, the slots are dynamic and thus such direct ID mapping cannot be used.

Therefore, within this protocol, the nodes are identified by a dynamic ID that corresponds to the ID of the slot they are assigned to ($0 \leq k \leq K-1$). A simple rule is used to map the static unique node ID to a dynamic slot ID. The currently lowest static ID among the running nodes is assigned slot 0, the following static ID is assigned to slot 1 and so on until the highest static ID among the running nodes that is assigned to slot $K-1$. This assignment is carried out every time there is a change in the slots structure, i.e., every time a node joins or leaves the group.

5.3.2 State machines to support joining and leaving

The dynamic insertion/removal of nodes is carried out in a distributed way based on the dissemination of the membership vectors. To manage this process, each node runs the state machine presented in Figure 5.12 for each of the other potential team members, i.e., considering the maximum possible number of nodes N. Thus, for each node, with respect to each other node, the implemented state machine has four states:

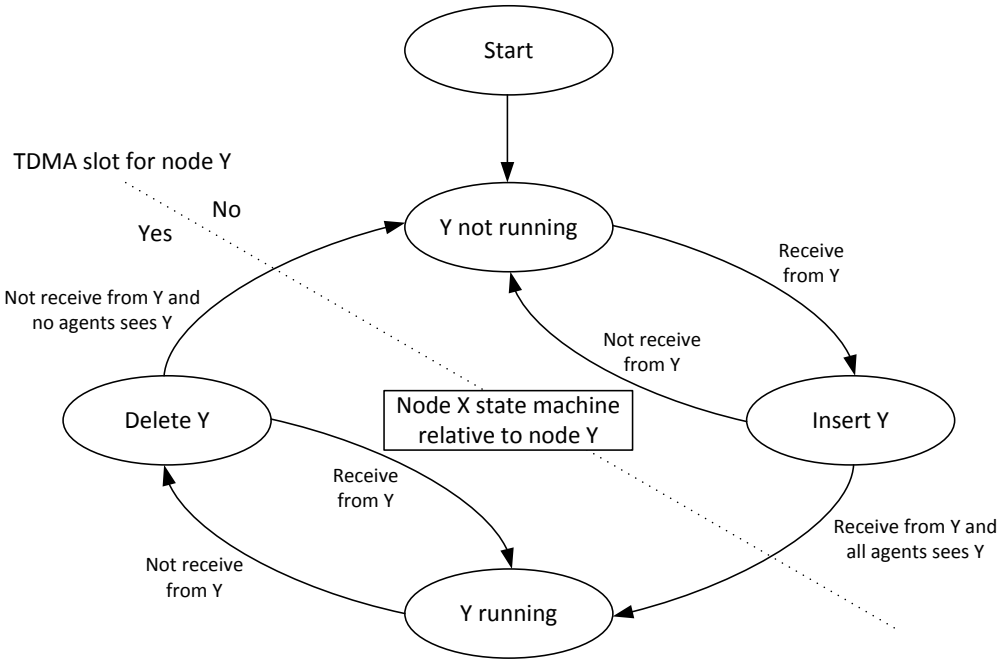


Figure 5.12: State-machine for capturing the state of another nodes

- **Not running** – The node is not powered up or is unreachable, i.e., not associated with the team wireless AP;

- **Insert** – The node started transmitting but has not yet been detected by all the current team mates. In this state the node has no slot yet in the TDMA round and thus it is transmitting out of phase as external traffic;

- **Running** – The node has been detected by all team members and its own slot in the TDMA round has been created. All nodes resynchronize and continue transmitting in their new slots;

- **Delete** – The node is not transmitting or its message was not received, e.g., due to an error.

On the other hand, the state machine of each node that manages its own internal state is rather simple (Fig. 5.13) having basically two states, only, **Insert** and **Running**. In fact, in our model a node never *deletes* itself or assumes to be *Not running*.

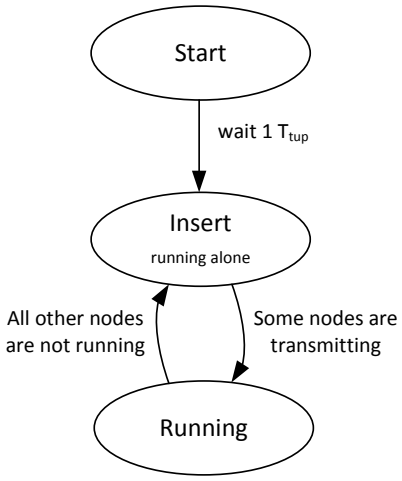


Figure 5.13: State-machine for the node itself

5.3.3 Operation of the Reconfigurable and Adaptive TDMA

When a new node arrives, it starts transmitting its periodic information in an unsynchronized way, i.e., as external traffic, with its own state as *Insert*. Meanwhile, all nodes, including the new one, continue updating their membership vectors with the received frames. During this initial period, which we call the *agreement* phase, the new node has no slot in the round, thus no dynamic ID, and the T_{xwin_K} value remains unchanged.

At the instant of its following transmission, one round later considering no packet losses or a few rounds later otherwise, the new node checks if all current team members have agreed on its existence, i.e., they all detected its transmissions and signalled it marking the new node in their membership vectors as *Insert*. This will eventually occur, leading to the end of the agreement phase and the start of the *reconfiguration* phase.

In the reconfiguration phase, the new node updates the number of active team members, K, and the slot duration, T_{xwin_K}, reassigns the dynamic IDs locally, computes the offset of its slot in the new round configuration, updates its state to *Running* and transmits its packet, still unsynchronized. Upon the reception of the next reference packet (dynamic ID 0), the slot offset is used to set a timer and trigger the new node transmission in its new slot thus concluding the integration of the new node.

However, the complete reconfiguration process only ends when all nodes adhere to the new round configuration. This is carried out node by node, as their transmission instants occur. Basically, every time a transmission instant fires, the respective team member checks whether any new node has joined during the past round and has been acknowledged by all other members, i.e, it already transmitted with *Running* status. If it has, then this node also reconfigures the round updating K and T_{xwin_K}, reassigning the dynamic IDs and computing the offset of its slot in the new round configuration. Then it transmits its packet, which is still in the slot of the previous configuration. Only after receiving the next reference packet, the new offset is used and the next transmission occurs in the new slot.

If this node did not receive the new node message with *Running* status in the previous round, it keeps the current round structure for one round more. Note that this can occur due to the phase adjustment of the transmissions of the new node when transiting to its new slot, but it can also occur due to packet losses, which simply cause an extension of the reconfiguration phase for an extra round.

The removal of an absent node uses a similar process. When in the previous m rounds, currently 10, no reception from a node is detected, the state of that node is changed to *Delete*. When all other running team members have also marked that node as *Delete* then the node is considered as *Not Running*, the number of active members K is decremented, the slot duration T_{xwin} is increased, the dynamic IDs reassigned and the slot offsets recomputed.

Figure 5.14 shows an example of the reconfiguration process of the TDMA protocol caused by the inclusion of a new node in the team. Note that the arrows denote the instants of transmission and the following blocks are the respective membership vectors, not representing the duration of the respective frame transmission in the wireless medium.

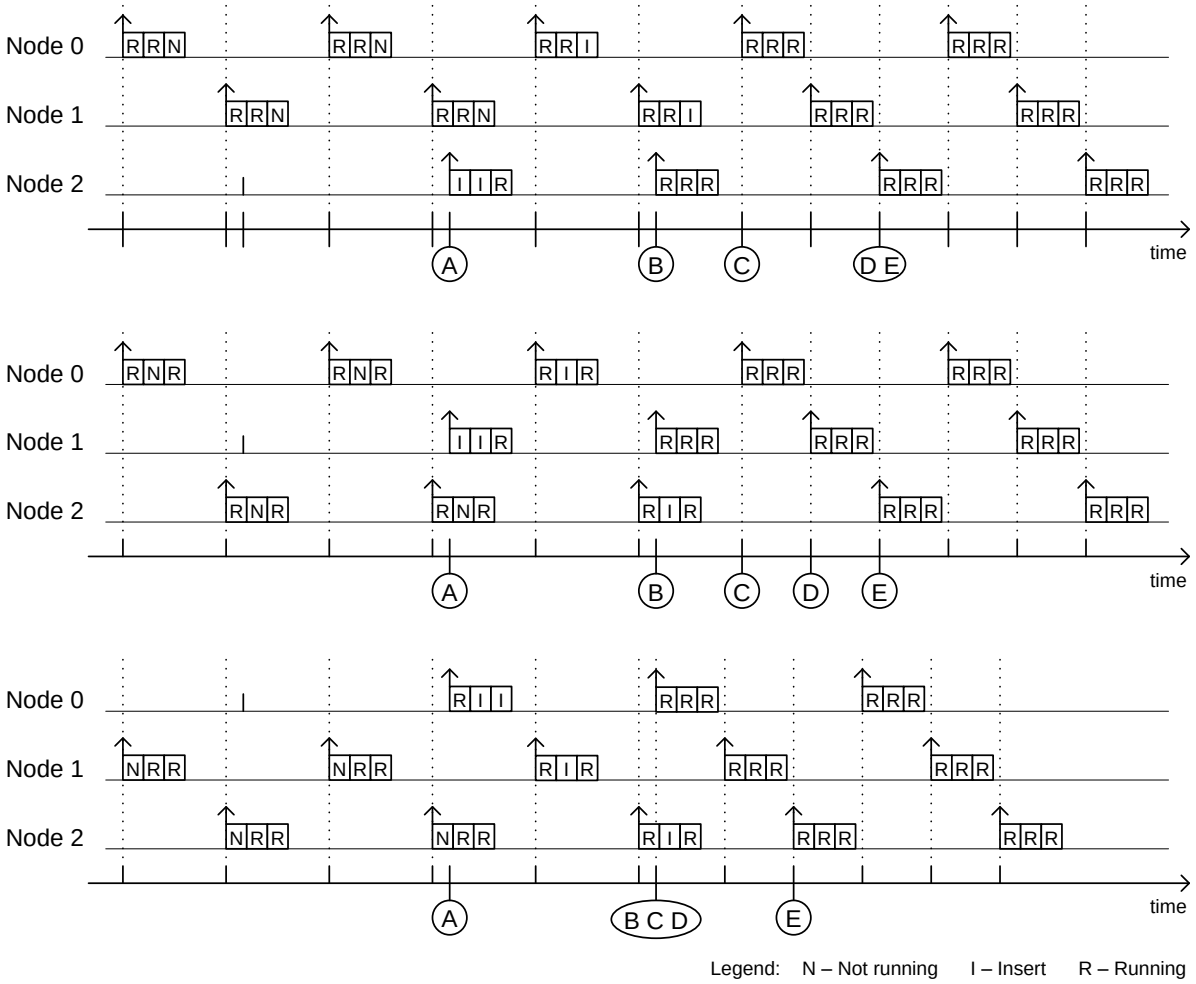


Figure 5.14: Timelines of three joining situations for adding a new team member
Top: joining member has highest ID; Middle: joining node has an intermediate ID;
Bottom: joining node has the lowest ID and will become the new reference

The upper timeline in Figure 5.14 represents a situation in which the joining node is the one with the highest physical ID. The agreement phase is delimited by A and B. The new node is integrated at D, which also corresponds to the end of the reconfiguration phase.

The middle timeline represents a situation in which the joining node has an intermediate physical node ID with respect to the nodes already in the team. In this case, the node will be integrated in the round and the reconfiguration phase ends one slot after.

Finally, the lower timeline represents a situation in which the joining node has the lowest physical ID with respect to the nodes already in the team and thus it will become the new reference. Note that once it changes its state to *Running*, it reconfigures the round internally

and immediately becomes the new reference node with dynamic ID 0. Thus, it is integrated as soon as the agreement phase ends, and all the other nodes will recompute their offsets with respect to its transmission. The reconfiguration phase ends with the last slot of this round.

The relevant points in the reconfiguration process timeline, namely A through E, are explained next.

- **A** – A new node has connected to the network and, after waiting T_{tup}, it starts transmitting, unsynchronized;
- **B** – After one round, the new node has received messages from all the other nodes with their membership vectors indicating the new node as *Insert*. Thus, the agreement phase is over and the reconfiguration phase is started. In its membership vector, it updates all states to *Running*, increments the number of team members to $K = 3$, updates the slot duration to $T_{xwin} = \frac{T_{tup}}{3}$, computes the offset of its slot and transmits its packet. Upon reception of this packet, all the other nodes update the state of the new node to *Running* and perform a similar round reconfiguration to 3 slots;
- **C** – The next reference packet is transmitted and received by all nodes. Upon this reception, each node sets up a timer to trigger the respective packet transmission in the right slot;
- **D** – The timer of the new node expires and this node transmits its packet in its newly allocated slot;
- **E** – The last node to transmit its packet in the new round configuration ends the reconfiguration phase.

Figure 5.15 shows the dynamic adjustment of the number of running nodes in a concrete scenario. In this case, there are initially six nodes running. Then a seventh node joins the team, with dynamic ID 3. Soon after, the node with dynamic ID 4 leaves the team. Later on, the node with dynamic ID 4 leaves the team for a while. While it is absent, the team is reconfigured to 5 running nodes. In particular, note the capacity of the protocol to maximize the time gap between the transmissions of the team members.

5.3.4 Time to join the team

As seen in the previous Section, we define the joining latency, T_{join}, as the interval since a new node starts transmitting, unsynchronized, until it is integrated in the team and starts transmitting in its own slot.

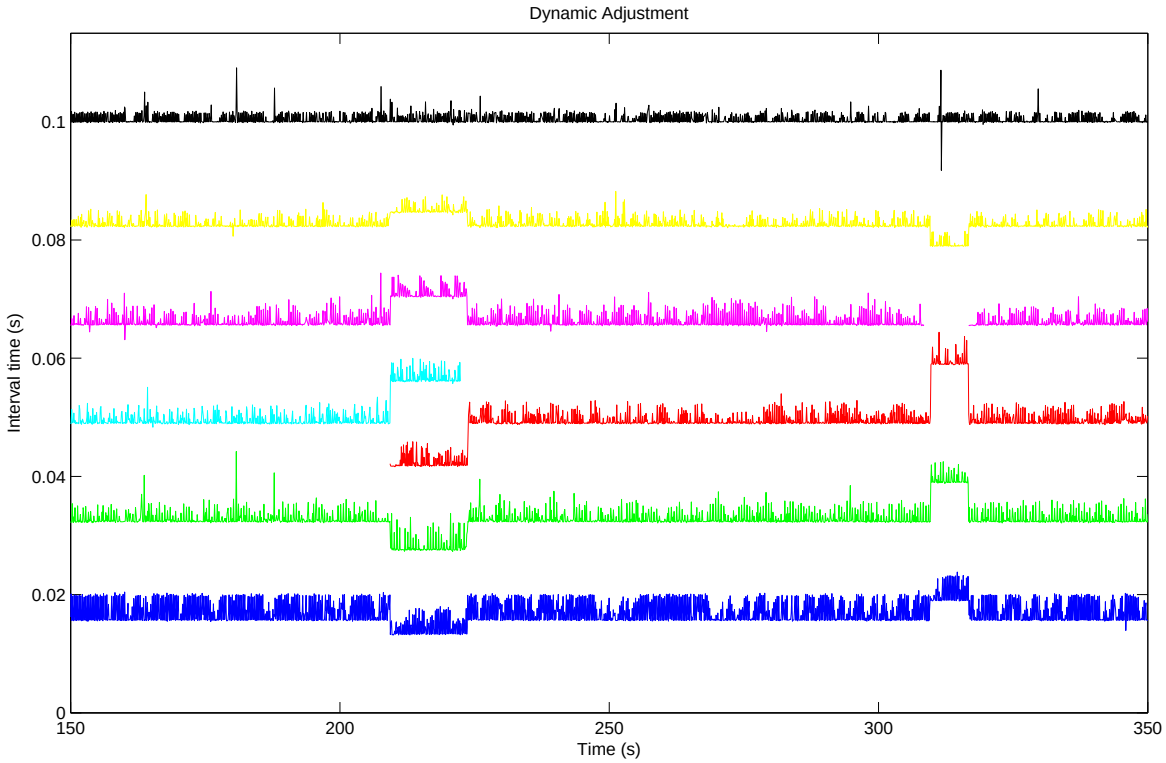


Figure 5.15: Membership vision of Agent 0 from other Agents over time

As explained, the joining process has two phases, the agreement and the round reconfiguration. Under normal operation, i.e., without packet losses, the first phase takes one round, as depicted in Figure 5.14. If there are packet losses affecting the team transmissions in this period, the agreement phase is extended another round, and possibly more rounds until it succeeds.

Then, the round reconfiguration phase starts with the new node already transmitting with status *Running*. This phase can be further divided in two intervals, until the next reference packet is received and from then on until the first transmission of the last node to transmit in its slot in the new round configuration. This is the total reconfiguration latency, which we will refer to as T_{reconf}. However, the joining latency T_{join} only accounts for the interval until the first transmission of the new node in its new slot.

The first of these intervals, i.e., until the reference packet is received, depends on the relative phase between the new node initial transmissions and the initial round. In the best case, it lasts the transmission time of the reference packet t_{packet} and in the worst-case, it takes one round plus the largest delay Δ_{K-1} that the reference packet might have suffered in this round, where K is the number of running nodes, including the joining one. In the presence of packet losses, this interval can be further extended by one or more integer rounds.

The second interval is given by the offset of the new node slot, i.e., $i \times T_{xwin}$ where i is the dynamic ID of the new node in the new round configuration. Therefore, T_{join} can be bounded by Eq. 5.13, without considering packet losses.

$$T_{tup} + t_{packet} \leq T_{join} \leq 2 \times T_{tup} + \Delta_{K-1} + i \times T_{xwin} \tag{5.13}$$

When considering the impact of errors and consequent packet losses, the upper bound needs to include the extra rounds incurred as given by Eq. 5.14 where $n(b, p)$ is the number of extra rounds that need to be considered for a given probability p of successful packet reception and given a bit error rate b.

$$T_{join} \leq (2 + n(b, p)) \times T_{tup} + \Delta_{K-1} + i \times T_{xwin_K} \tag{5.14}$$

Similarly, the reconfiguration latency can be upper bounded by Eq. 5.15.

$$T_{reconf} \leq (2 + n(b, p)) \times T_{tup} + \Delta_{K-1} + (K - 1) \times T_{xwin_K} \tag{5.15}$$

Note, however, that the analytic model of the $n(b, p)$ function was not considered in this work. In Chapter 7 we show an experimental estimation of the distribution of this function.

5.3.5 Adding multiple slots per node

In certain use cases with teams of robots, different nodes can have different real-time constraints, some requiring higher reactivity than others as imposed by diverse collaborative behaviors. For example, a robot could be equipped with a special sensor that would require an update rate of $25ms$ while the other team members would just require a rate of $100ms$ to share their own state.

In this case, we can setup a round of $25ms$ and then have the remaining (*slower*) robots transmitting four times more often than needed, imposing an overhead penalty. Alternatively, the *slower* robots could transmit only once every four rounds, but this would then impact the team synchronization negatively.

Another option is to setup a round of $100ms$ and have the *faster* robot transmitting four times in each round. This option is preferable from an overhead and synchronization points of view but its implementation in a dynamic round structure is not trivial since it is important to guarantee that the additional transmissions inside the round occur separated as close to $25ms$ as possible, even during round reconfigurations.

To implement this feature we use two mechanisms. On one hand, we allow each node to define several physical IDs instead of just one. The protocol interprets these physical IDs as different nodes and thus creates the desired extra slots that the node needs. Then we developed a more elaborate mapping of physical IDs to slots that forces the multiple slots of each robot to be separated from each other as much as possible in the round phase space.

This mapping is achieved with Algorithm 1, in which M is the total number of slots required in the round, computed dynamically (lines 1-5), and $reqSlots$ is the number of required slots per node. The slots of a node i are now identified by a duplet (i, j) where j is an associated index. Moreover, $\lfloor x \rceil$ represents rounding x to the nearest integer and $freeSlotNeighbor()$ is a function that returns the next free slot.

Algorithm 1 Allocation of multiple slots per node

```
1:  M ← 0
2:  for i ← each node do
3:     reqSlots[i] ← # required slots for node[i]
4:     M ← M + reqSlots[i]
5:  end for
6:  for i ← each node in descendant order of required slots do
7:     firstFree ← first slot free
8:     slotSpace ← M/reqSlots[i]
9:     for j = 0 to reqSlots[i] − 1 do
10:       s = firstFree + ⌊j × slotSpace⌉
11:       if slot[s] is not free then
12:          s = freeSlotNeighbor(s)
13:       end if
14:       slot[s] = #id(i, j)
15:    end for
16: end for
```

Line 8 computes the optimal distance between consecutive slots of the same node while the cycle of lines 9 to 15 assigns IDs of the node to slots with such separation. If one slot is busy (line 12), the next free slot is taken. This creates a deviation to the regular spacing between the node slots. Therefore, to minimize the impact of such deviation, the outer cycle (lines 6 to 16) that goes through all nodes starts from those that require more slots, which tolerate less these deviations, to those that require less slots, which tolerate larger deviations.

All remaining protocol structures and control variables, such as the membership vector, are now defined in terms of the maximum number of slots, which are taken as virtual nodes.

5.4 Summary

This chapter presents the Reconfigurable and Adaptive TDMA protocol, to be used over the IEEE 802.11 standard for wireless communication.

The described protocol allows to periodically share the state data of each node to the remaining members of the robotic team, using a method similar to the TDMA, without the need of a global clock synchronization. The main advantages over the common TDMA method are:

- Copes with the presence of external non-controlled traffic (channel sharing);
- Adapts the update period to the external interferences;
- Dynamically reconfigures according to the number of team members.

Chapter 6

Real-Time Database

Effective collaboration among robots requires data exchange, for example, to accomplish coordinated tasks or to carry out sensor fusion to improve and/or extend the knowledge about the surrounding environment.

The adoption of a middleware, as discussed in Chapter 3, allows accelerating the process of develop collaborative behaviors hiding the complexities associated to data exchange between processes, particularly across different nodes.

In this chapter we present a simple middleware, the RTDB, that provides an abstraction of remote data presenting it as if it is local, using proxies. These proxies are updated in the background, transparently to the user applications that access the data, at a rate that ensures its temporal validity. Moreover, the proxies also provide age information together with every data item, which allows application processes to use temporal models of the respective processes to estimate the current value of the real entity at the time it is read from the proxy.

This allows a separation between the network data exchanges and the execution of application processes that is rather beneficial for the latter, in terms of temporal behavior. It is also a rather simple abstraction to support construction of collaborative behaviors, contributing to speed up their development. For this reason, this middleware has generated some interest among robotics groups and was used in several contexts, particularly in teams of the MSL of RoboCup.

6.1 Architecture

The architecture of the RTDB is based on that of the classical Blackboard architecture [17, 41], developed in the early days of the artificial intelligence field and still widely adopted in many applications. A Blackboard is a public repository of information where multiple processes publish their data. The repository is a shared resource, that can be local or remote, where all the processes can read and write data. The Blackboard can include raw input data, partial and final solutions, and control information. It also acts as a communication medium and buffer. [17] discuss similarities and complementarities between Blackboard-based systems and multi-agent systems.

The RTDB uses a similar approach to the Blackboard with the extension to provide interfaces to multiple machines (agents) with data proxies for fast access. This is accomplished replicating in all the agents the data that each one shares, resulting in a data structure in each node that encompasses local data together with copies (proxies) of shared remote data. A specific communication manager refreshes the RTDB contents, see Figure 6.1, ensuring consistency between the original data entities and their proxies. This way, all the processes running on an agent can access remote data items as if they were local, thus in a fast way, without communication delays.

The replication of the contents shared by one agent in the RTDB to the other agents implies the utilization of network resources that can be subject to several constraints as with wireless communication (Chapter 2). Noting that not all the information generated by an agent needs to be shared with others, for instance data used in local inter-process communication or temporary data, the volume of data to replicate can be small. Thus, the RTDB is internally divided in two distinct memory areas:

- **Local** - Used to hold the data that is only relevant to local processes and will not be broadcast to the other agents;
- **Shared** - That keeps the data that is relevant for cooperative behaviors and thus will be broadcast. This area is organized in blocks, one dedicated to each agent. In particular, one block contains the data that the holding agent shares, which will be broadcast by this agent, while the remaining blocks contain the data shared (broadcast) by the other agents.

Each process connects to the RTDB through the local RTDB API that provides the necessary methods to access the data, transparently to the block in which the data actually resides inside the RTDB.

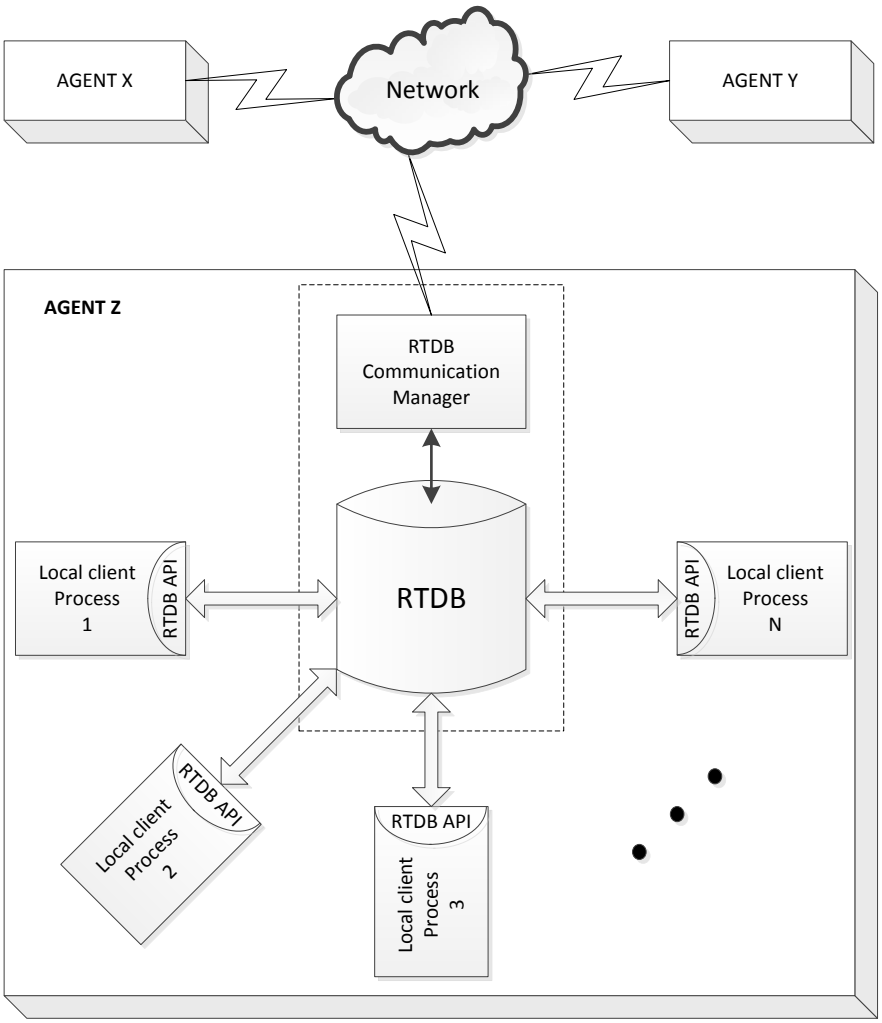


Figure 6.1: Agent-centered view of the RTDB architecture

The heterogeneity of agents generates different requirements for memory usage and consequently different RTDB block sizes, see Figure 6.2. The size of each block and the choice of data that must be broadcast or kept locally is defined a priori through a configuration file.

As mentioned above, the access to remote data is done without explicit use of communication, abstracting away the data distribution itself. The refreshment of the remote data is carried out in the background by the communication manager that must consider the specific temporal validity of the data items, the constraints of the communication medium and the amount of data to exchange. A communication manager that does cyclic refreshment of the RTDB shared areas at an adequate rate is the subject of Chapter 5. However, note that other refreshing policies and protocols are possible. In fact, there is a complete separation between the communication protocol used for RTDB refreshing and the RTDB itself.

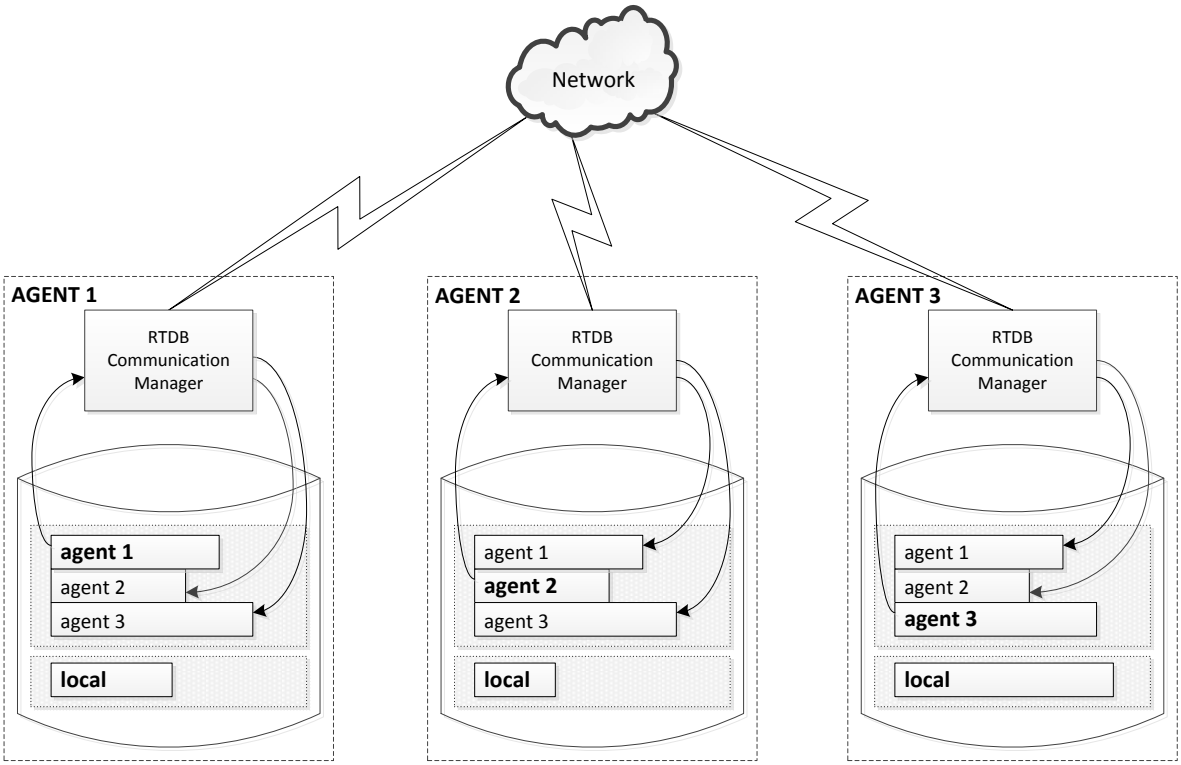


Figure 6.2: Network view of the RTDB architecture

It is also possible to use the RTDB for local inter-process communication, only, within a single agent. In fact, it has been successfully used in such configuration, for example, in the CAMBADA@Home [7] and ROTA [57, 61] projects. In this case, the RTDB becomes a local Blackboard, just with the local block, but still extended with additional information on age, which is always present, either in local or distributed implementations as explained next.

Note that, despite the fast local access to data provided by the RTDB to the local processes in each agent, the data itself always has a certain age (analyzed in detail in Section 6.7). This age represents an estimate of the time elapsed since the original data item was written to the RTDB by the producer process at the source agent until it is retrieved by a consumer process in a remote agent.

The knowledge of this age can be of great importance to the programmers of collaborative behaviors. For example, it allows detecting stale data, i.e., data that is too old to be useful. It can also be used with temporal models to predict the actual value of that item at the time it is consumed given the value it had at the time it was produced. Therefore, the RTDB keeps a field for each data item indicating the age of the data since it was written for the last time. This age is provided to the users through the API methods as we will see in Section 6.4.

The design of the RTDB architecture was motivated by the problems identified in Chapter 4, found in the MSL of RoboCup. However, it was devised using a generic approach, so that it could be used in different applications. Basically, we believe it is a useful component to develop teams of cooperative autonomous robots that collaborate through state sharing.

6.2 Configuration

In order to be used the RTDB must be configured adequately. In its current form, the RTDB is a static component with a structure defined offline based on knowledge of the specific team characteristics and the data to be included.

The configuration of the RTDB is done automatically by parsing a configuration text file that specifies the team characteristics and the RTDB composition. As mentioned previously, the team can be composed of multiple agents with different roles and equipped with different sensors and actuators, thus having distinct data requirements for either local and global communication. Hence, the configuration file allows describing each agent from the data point of view and altogether represents the team model. Listing 6.1 shows the model used in the RTDB configuration.

```
AGENTS = <<id_ag>> [, <<id_ag>> , ...] [;]

ITEM <<id_it>> { datatype = type; [headerfile = <<filename>>]; [period = <<number>>]; }
...

SCHEMA <<id_sc>> { [shared = <<id_it>> [ , <<id_it>>, ...] ; ]
                [local = <<id_it>> [, <<id_it>>, ...] ; ]
...

ASSIGNMENT { schema = <<id_sc>>; agents = <<id_ag>>, ... ; }
...
```

Listing 6.1: The RTDB configuration model

The meaning of each of the constructs is explained next:

- `AGENTS` – Set of agent unique identifiers that specify the agents that compose the team. Each agent identifier (`id_ag`) will also be used in the actual application code when accessing remote data in the RTDB to specify which agent to retrieve the data from;
- `ITEM` – This is a data unit kept in the RTDB and handled as an integer piece of information. Each such item is identified by a unique identifier (`id_it`) together with the following three attributes:

  - `datatype` – This is the actual type of the item data and it is used to compute the data size in bytes, necessary to hold the item in memory. It can be a predefined type, such as `int` or `double` , or a user defined type;
  - `headerfile` – This attribute is needed if the data type is user defined, only. It contains the path and name of the `C language` header file where the data type is defined;
  - `period` – Sets the item refresh period in multiples of the communication cycle. This allows adjusting the RTDB communication requirements to the actual dynamics of its items. It is used in shared items, only.

- `SCHEMA` – Set of local and shared `ITEMS` produced by a given agent type. Each schema has a unique identifier (`id_sc`). There can be one or more schemas. The `ITEMS` are thus specified in two lists, accordingly:
  - `shared` – The list of `ITEMS` to be shared, i.e., broadcast;
  - `local` – The list of `ITEMS` that are available to local processes, only;
- `ASSIGNMENT` – Associates one `SCHEMA` to one or more `AGENTS` and thus allows defining all RTDB instances. There can be one or more assignments depending on the number of different agent types.

To better explain how the configuration file is used, Listing 6.2 shows an example of a configuration for a team of three `AGENTS` with two similar mobile robots (`robot1` and `robot2`), that explore the environment using a camera, and a base station (`base`), that is responsible for data fusion and world model construction.

```
AGENTS = robot1 , robot2 , base ;

ITEM image {datatype = struct image ; headerfile = image.h ; }
ITEM position { datatype = struct pos ; headerfile = pos.h ; period = 1 ; }
ITEM obstacles { datatype = struct obstacles ; headerfile = obstacles.h ;
                 period = 1 ; }
ITEM fuse_data {datatype = struct fuse ; headerfile = fuse.h ; period = 1; }

SCHEMA robot { shared = position, obstacles ;
               local = image ; }
SCHEMA base_st { shared = fuse_data ; }

ASSIGNMENT { schema = robot ; agents = robot1 , robot2 ; }
ASSIGNMENT { schema = base_st ; agents = base ; }
```

Listing 6.2: Example of an RTDB configuration file

The image acquisition processes running on the robots save the raw images data in the local item `image`. The self-localization and obstacle detection processes read the `image` item and save the results of self-position and obstacles localization in the shared items `position` and `obstacles`, respectively. These two items will be broadcast to the other agents each communication cycle.

A third node (`base`) equipped with a powerful computing system, not necessarily a mobile robot, is responsible for constructing the world model of the environment, combining the data received from the robots. This world model is then shared with the robots, for example, allowing them to choose areas that are still to be explored, improving the efficiency of a collaborative SLAM approach.

This example shows the capacity of the RTDB middleware to support collaborative applications among heterogeneous agents. However, the team composition and data requirements must be known a priori, when the code of the agents is compiled, and thus cannot be changed during execution. Note, nevertheless, that a team can be configured according to its maximum dimension and requirements. Then, at run time, the actual number of working robots can be less but the collaborative applications need to be prepared for this possibility.

6.3 Internal Structure

As shown in Figure 6.2, the RTDB is formed by a set of shared memory blocks, one of which is used only for local data, while the other ones are used for data distribution. Physically, these blocks are implemented in two areas each, one containing a set of records that are needed for data control purposes and the other holding the actual data (Figure 6.3). In turn, the data area is divided in two banks (`bank[0]` and `bank[1]`) implementing a double buffer reader-writer synchronization per item, as explained in Section 6.5.

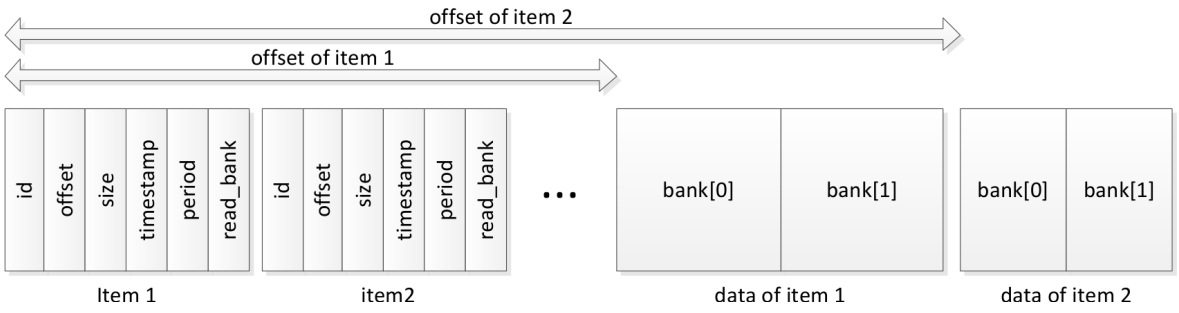


Figure 6.3: The internal organization of the RTDB blocks in control records and associated data

Each record describes an item, containing the fields referred in Listing 6.3, specified in the `C language`. These are: an internal item identifier `id` that is a sequential number generated by the parser, corresponding to the item identifier `id_it` in the configuration file; an `offset` from the beginning of the shared memory block to the data area of the respective item, as shown in Figure 6.3; the `size` in bytes of the item data; the `update` period, as defined in the configuration file, reflecting the dynamics of the respective item; a `timestamp` with the local time of the last write operation at each bank, for computing the age of the data; and, the `read_bank` control field used for managing the double buffering.

```
typedef struct {
  int id;
  int offset;
  int size;
  int period;
  struct timeval timestamp[2];
  int read_bank;
} TRec;
```

Listing 6.3: The fields of a generic RTDB record

6.4 RTDB API

The RTDB is fully implemented in `C language`. The functionality of the RTDB is available through a very simple API with only four methods, as shown in Listing 6.4. Two additional methods are used internally for updating the remote items.

```
public:
  int DB_init (void)
  void DB_free (void)
  int DB_put (int id_it, void *data)
  int DB_get (int id_ag, int id_it, void *data)

protected:
  int DB_comm_init(RTDBconf_var *rec)
  int DB_comm_put(int id_ag, int id_it, int size, void *data, int age);
```

Listing 6.4: The RTDB interface methods

The `DB_init` method is called once by every process that needs access to the RTDB and handles initialization issues. The actual memory allocation for holding the RTDB in each agent is executed by the first process to invoke such call. Subsequent calls just increment an

internal process counter with the number of processes that are currently linked to the RTDB. Conversely, the method `DB_free` does the corresponding clean up and detaches a process from the RTDB. It decrements the process counter and, when zero, frees the respective memory blocks. Both `DB_init` and `DB_free` return the value `0` upon successful execution or `-1` in case an error occurs.

The actual access to the RTDB memory areas is carried out with the non-blocking methods `DB_put` and `DB_get`. Both methods use the item identifier `id_it` defined in the configuration file as well as a pointer to the data to be written to or read from the RTDB, respectively. `DB_get` further requires the specification of the agent `id_ag` from which the item to be read belongs to, which is used to identify the respective data area in the database. This method returns the age of the data retrieved from the RTDB in milliseconds or `-1` in case of error. The method `DB_put` returns the total number of bytes copied to the RTDB or `-1` indicating an error. For consistency purposes, as explained in Section 6.5, a writer process can only invoke `DB_put` once per RTDB item in each execution cycle.

The access to items defined as local or shared is transparent since the item identifier `id_it` is unique for all the items saved in the RTDB. This allows transforming a local item to shared, or vice-versa, with a simple change in the configuration file, simply moving the that item across the respective lists. This can be very useful for debugging purposes, to have temporary access to agents' local data at run time, in a monitoring station.

Finally, the RTDB API includes two other methods that are protected (`DB_comm_init` and `DB_comm_put`) and thus, not available to an ordinary user process. These are used exclusively by the communications manager in each agent.

With respect to the RTDB access, the communications manager is rather similar to an ordinary user process, also making use of the same `DB_init`, `DB_free` and `DB_get` methods to begin and finish the access to the RTDB and to read data from the RTDB shared areas. However, it uses `DB_comm_init`, invoked once when the process is launched, to retrieve communications relevant information from the RTDB, such as `period` and `size` for all items that are to be broadcast by this agent, allowing to compute an internal transmissions schedule that determines which items to broadcast in each communications cycle.

On the other hand, the `DB_comm_put` method is used to write in the shared RTDB areas the remote data received through the communications interface. The `size` parameter is used for a simple validation of the received data, comparing the received data size with the expected data size. The `age` parameter is the age of the received data at the reception instant, thus including the producer and transmission components of the age. This is an age offset that will be written in the RTDB that will allow computing the total data age at the time of data consumption, by the `DB_get` method, as presented in Section 6.7.

6.5 Synchronization of concurrent read/write accesses

The underlying data sharing model of the RTDB consists of concurrent writer processes that generate data and reader processes that consume it. However, each data item has only a single writer process while it can be read by multiple reader processes. This is a typical single writer multiple readers synchronization case. In this situation, the multiple readers can access concurrently each data item freely, without access control. On the other hand, concurrent accesses between the writer process and any of the reader processes need to be controlled to avoid data corruption.

This is a well known synchronization situation in the access to shared data which, under certain circumstances explained further on, can be solved with a double buffering technique. In the RTDB we make use of this technique due to its simplicity and reduced blocking, despite an extra cost in memory usage.

6.5.1 Using single buffer synchronization

An initially developed version of the RTDB used a single buffer technique [49]. However, this technique requires that the writer has higher priority than the readers in case of a preemptive system. The readers need to use a consistency flag that is set by the writer whenever it interrupts a reading operation. The readers use this flag at the end of the reading operation and repeat the reading if the flag is set.

In our case, the higher priority of the writer is easy to enforce in the network-to-RTDB transfers by giving the communications manager higher priority then any other user process. In the RTDB-to-network transfers, we would need to give the communications manager a priority lower than all user processes. This could be achieved separating the communications manager in two processes, to handle the transfers in each way, with different priorities, or using a single process but changing priority dynamically. However, giving the communications manager a low priority decreases the control on its transmission instants, which is undesirable from the communications scheduling point of view. Moreover, in the local communications it would be impossible in the general case to give the writer higher priority than any reader, as many processes would be writers and readers.

An alternative to maintain the single buffer approach would be to disable preemption during buffer accesses, either for writing or reading. For short data items, this is an effective solution, but when the data items are large, this technique may also imply a significant blocking of the respective processes, which is also undesirable for the communications manager.

However, in complex operating systems, e.g., Linux, disabling preemption or assigning a higher priority level requires administrator (root) rights. Carrying out these operations within user level processes is a path to bugs that may compromise system robustness as it exposes the core system integrity to user programming errors. Therefore, the single buffer synchronization approach was excluded leading to the option for double buffering. We still use higher priority in the communications manager since it is a system component and it is relevant to reduce its execution blocking to achieve better timeliness. Nevertheless, all user processes run exclusively at user level, without need for root execution rights and independently of the scheduling policy in place.

Finally, the option for a priority-independent solution also drifts our work away from other works in the general field of real-time databases and services [82] that aim at ensuring the timeliness of the read and write transactions and the freshness of the data through appropriate scheduling techniques typically based on priority management.

6.5.2 Using double buffering synchronization

The double buffers are implemented with the two banks per item shown in Figure 6.3. At each moment, all the reader processes of a given item read from the same bank indicated by the `read_bank` field in the item control record. When the respective writer process wishes to write a new value, it checks the bank currently in use by the reader processes, i.e., indicated by the `read_bank` field, and saves the data in the other bank. When it finishes, it updates the `read_bank` field to point to the bank with the new data. This way, the readers will always fetch from the bank with the most fresh data.

Nevertheless, under concurrent preemptive execution, it is important to analyze potential race conditions involving the writer and reader processes. In general, given the use of different banks for the writer process on one hand and the reader processes on the other, there is no consistency problem. The situation that raises more concerns is when the writer process preempts a reader, since the former updates the `read_bank` field while the latter is still reading. This situation is shown in Figure 6.4. However, the reader continues reading from the same bank until the end of that instance and there is no data corruption. As a drawback, the data retrieved by the reader is not the most fresh, but using the data age associated with each bank, the reader process can adopt a reactive procedure and, for example, carry out a new reading if beneficial.

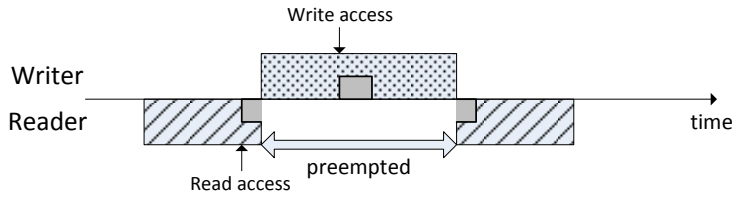


Figure 6.4: Concurrent access to the same RTDB item

The absence of data corruption in the situation shown in Figure 6.4 holds as long as the writer does not issue consecutive write operations while preempting the reading. In such case, the first write would change the `read_bank` at the end of its execution to point to the bank in use by the reader and a second write would overwrite that bank corrupting the reader data.

Nevertheless, consecutive writes on the same data item by the same process during one execution instance should not occur, representing an erroneous design pattern. A correct pattern of each process execution instance starts with data reading, followed by data processing and ending with writing the results, once per item and per execution cycle.

Another situation that could generate problems is one in which the writer, despite issuing a single write operation in each instance, would be quickly reactived so that it could preempt the reading operation twice. Again, this situation cannot occur in our setting since the period of the processes used is in the order of a few tens of miliseconds, which is much longer than the time to access the data items.

Therefore, the presented double buffering technique is safe in our execution context.

6.6 RTDB replication management

Figure 6.2 provides a network view of the RTDB architecture, highlighting the replication of its shared components in all nodes. Managing this replication is relatively easy in our approach because of two aspects.

Firstly, there is a single writer for each shared item, which avoids complicated arbitration mechanisms that would be needed if there were multiple writers and would necessarily increase the potential for inconsistency.

Secondly, we use state semantics in the RTDB contents, which is more tolerant to small inconsistencies during item updates than if an event semantics was used. In fact, the contents of the RTDB represent samplings of continuous time signals. If, during an update operation, one inconsistency occurs, for example, caused by a local communication error, this inconsistency will only last until the next successful update, normally one item communication

period later. Moreover, such inconsistency may represent just a small inaccuracy in the local signal representation and can be partially mitigated using the age information and temporal models of the items dynamics, or even control approaches that cope with errors and vacant sampling.

The RTDB database management system is embedded in the communications manager. In fact, it is this component that controls the dissemination of information and thus, the updating of the remote proxies of each shared item. As explained in Chapter 5, this updating is carried out with broadcast communication, thus making an effective use of the network bandwidth and further contributing to the global RTDB consistency.

One important aspect is the clear separation between the RTDB itself and its database management system, i.e., its communications manager. This separation was already referred in a different context but it is also relevant here. For example, it is possible to use the RTDB with an event semantics simply by changing the communications manager adequately. In such case, an event would trigger an update of the respective item. However, this approach would need reliable communications, with acknowledging mechanisms, to prevent lasting inconsistencies to occur.

6.7 Age of data

Knowing the age of the data can be very useful for collaborative behaviors to detect and possibly mitigate situations of loss of temporal validity. However, for the sake of simplicity, our middleware does not include a global clock service implying that the clock in each robot is not correlated. To circumvent such difficulty, the middleware computes time intervals, only.

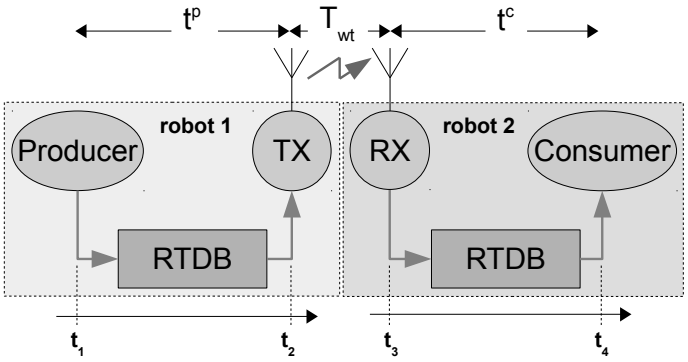


Figure 6.5: Datum age calculation

When a producer writes an item in the RTDB, the local time $t1$ is saved in the `timestamp` field of the item record, as shown in Figure 6.5). Later on, when the communications manager fetches the data to disseminate it to the other agents, it computes for each item to be transmitted the difference between the current local time $t2$ and the saved $t1$, which is the age of each datum at the time of transmission in the producer side. The datum age at the producer side $t^p = t2 - t1$ is attached to each datum itself and transmitted together in the network packet.

When the packet is received by the communications manager at the consumer side, each item is individually written in the RTDB shared area that corresponds to its producer. The data age received from the producer is subtracted from the current local time at the consumer, $t3$, and the result, $t3 - t^p$, is saved in `timestamp`. When a consumer process retrieves the item from the RTDB, the difference from the current time to the value saved in `timestamp` is computed, resulting is an estimate of the age of the data, from the moment it was produced to the moment in which it was consumed.

This estimation, however, still lacks the transmission time (T_{wt}), which depends on the actual bit rate, on the latency to access the medium and on possible re-transmissions. However, as shown in Chapter 7, the communication protocol that we use together with this middleware (Chapter 5) has a positive impact on the transmission time, leading to a relatively constant latency that can be easily added to the age estimation to improve its accuracy.

6.7.1 Upper bounding the age of data

The communication protocol we currently use (see Chapter 5) is not synchronized with the control system of the robots, due to the adaptive nature of the protocol that keeps changing its cycle duration. This may lead to extra delays in the refreshing of the remote data that the programmer must be aware of.

In particular, when a robot accesses a local image of a datum from another team member, that datum can be as old as:

$$max\_data\_age = min(T_{rcpp}, n_{dup} * T_{tup}) + T_{wt} + (n_{dup} * T_{tup}) \tag{6.1}$$

This worst case data age corresponds to when the communications manager fetches the data in the RTDB for transmission just before that data being updated by the respective producer process in the respective robot. Thus, at that point, that data can be as old as one period of the respective producer (T_{rcpp}).

However, this latency cannot be larger than the item update period configured in the RTDB ($n_{dup} * T_{tup}$) thus, the minimum of the two must be considered. Note that n_{dup} is the refresh period in integer number of communication cycles defined in the item control record, and T_{tup} is the communication cycle duration, defined in Chapter 5 as the Team Update Period.

The transmission of the data over the air takes some time that must also be accounted for (T_{wt}).

Finally, when the consumer accesses the data on its side, the data can be waiting in the respective item buffer for at most another item update period ($n_{dup} \times T_{tup}$).

Within the above expression, only the wireless transmission delay is unknown and may vary with the traffic load in the network, requiring an adequate estimation.

The minimum age of any datum corresponds to the situation in which the transmission takes place right after the producer updated the item and the consumer accesses the item right after it has been received, being thus given by

$$min\_data\_age = T_{wt} \tag{6.2}$$

This large difference between maximum and minimum age shows that the item age can be affected by high jitter as is typical in situations in which items are propagated through unsynchronized cycles. The order of magnitude of the max_data_age determines the dynamics of the collaborative behaviors that this database management system can cope with.

6.8 Scheduling the dissemination of RTDB items

In a state based approach, as we are currently following, the update of remote shared items is done cyclically, controlled by the communications manager. As mentioned above, the basic communications cycle is T_{tup} and each item becomes ready to be shared every n_{dup} cycles. This parameter is specified individually per item and allows reducing the communication load whenever some of the items have lower dynamics than the communications cycle and thus can use a higher n_{dup} parameter. Moreover, the communications manager also encapsulates multiple items in the same network packet to reduce communications overhead.

When all the shared items in the RTDB have $n_{dup} = 1$, then they are scheduled for transmission every communication cycle and there is no need for additional scheduling mechanisms. However, when the RTDB contains items with $n_{dup} > 1$ then there is need to carry out additional items scheduling, for example, to balance the communication load generated by each agent from cycle to cycle.

Currently, we specify at configuration time a given total payload in bytes (D_i) available for each of the N possible agents ($i = 0..N-1$) to transmit RTDB items per cycle. This total payload may involve several network packets and may be different for each agent, depending on the individual communication requirements.

The items scheduling model that we use is rather simplified. It takes into account, for a generic item k in agent i its size in bytes (C_i^k) and its period in number of communication cycles $n_{dup,i}^k$. Then we follow a Rate Monotonic approach according to which we schedule the items with shorter periods first, considering all of them ready at RTDB start time, and implicit relative deadlines equal to their periods.

In each cycle, the communications manager schedules ready items until $\sum_l (C_i^l)$ becomes as close as possible but below D_i, where l is an index to the ready packets already scheduled in that cycle. At that moment, any pending ready packets are left to be scheduled in the following cycles.

The maximum value of D_i must be set so that the respective transmission time is sufficiently below the minimum slot time $T_{xwin} = T_{tup}/N$ so that there is time free for extra load in the medium as well as for retransmissions upon error. Following the same terminology and reasoning of Eq. 5.3 in Chapter 5, we can express the upper bound on D_i as in Eq. 6.3.

$$D_i \leq \bar{D_i} : \frac{nodeLoadTime(\bar{D_i}, frameType, netType)}{\Omega - extLoadOcup(L, frameType, netType)} \leq T_{xwin} \tag{6.3}$$

However, in order to balance the communication load imposed by agent i in all communication cycles, we may be interested in determining which is the minimum value of D_i that allows transmitting all its items within the respective periods. This can be achieved using a server design technique from the hierarchical real-time scheduling theory [5, 11], in which the server capacity in bytes is given by D_i corresponding to one or more packets transmitted non-preemptively with a maximum packet length of Pkt. This model is also similar to one kind of limited preemption proposed recently in single processor scheduling to improve schedulability [14].

Finally, to determine the right packet size Pkt it is important to balance both communication overhead and robustness. Longer network packets incur in lower overhead but also higher probability of corruption by errors. Therefore, average packet lengths typically represent a better compromise that depends strongly on the bit error rate of the specific operational environment. Lower bit error rates allow longer packets with overhead reduction benefits. Computing the maximum packet size for a given reliability target is beyond the scope of this work. In our current work Pkt is set empirically.

6.9 Summary

This chapter presented a distributed shared memory middleware named RTDB that follows a similar approach to the typical Blackboard but enhanced to reduce communications and data access times, and to provide information on data age. It is particularly suited to support global state sharing among a team of autonomous mobile agents/robots.

We described the RTDB architecture, configuration model and the programming interface of the RTDB. Then we addressed the reader/writer synchronization problem and explained the option for a double buffer mechanism. This chapter also discussed the replication management approach, based on a dissemination protocol that updates remote items transparently to the applications, in the background, with a frequency adapted to the dynamics of each item.

This chapter also explained how the information on age is computed and maintained. This is one of the main features of this middleware that allows detecting stale data as well as estimating the current value of data items using models of their dynamics.

Comparing the RTDB with the middlewares presented in Chapter 3, it is considerably simpler exploring the fact that common applications of collaborative robotics frequently use a team of known robots, with a priori known features, and with a known maximum number of robots. This allowed using a static RTDB, leaving the support for run-time addition of new agents or new data items for future work.

Another option taken towards favoring simplicity was the use of a memory copy type of interaction. This explores the fact that, frequently, teams of robots have a similar computing architecture. This limitation can be easily mitigated using abstract data types inside the RTDB and it was also left for future work.

Chapter 7

Experiments

This chapter presents experimental results that aim at validating the claimed properties of the Reconfigurable and Adaptive TDMA protocol. These, in turn, have a direct impact on the consistency and timeliness of the RTDB middleware and thus on the performance of the collaborative applications that run on top.

We start by showing the benefits of using a synchronized TDMA approach in the communications for a team of cooperating autonomous agents. We show benefits on both transmission latency and packet losses. Then, we also show a thorough comparison with a traditional TDMA implementation based on clock synchronization. The results show the desired effect of our adaptive approach on increasing the resilience to coherent periodic interferences, i.e., those with a similar period or with a period that is close to an integer (sub)multiple. However, the advantage of our approach with respect to the clock synchronized one seems to disappear in the presence of strong bursty external traffic. This is also expected as discussed further on.

Finally, we validate the protocol operation in real operational conditions, particularly in RoboCup MSL games. We show the control and correctness of the membership management system, particularly the integration and removal of nodes, as well as delays and packet losses. We end with an analysis of several temporal parameters, namely the effective round period and the time to join the team.

7.1 Experimental setup

In order to test the benefits of the Reconfigurable and Adaptive TDMA communication protocol, several experiments were conducted to assess the number of lost packets with and without additional traffic load, with and without the synchronization scheme and with two different synchronization approaches. The network was configured in infrastrutured mode, i.e., all transmissions were carried out through the AP.

We used an experimental setup in a laboratory comprising four nodes and a monitoring station that time-stamped and logged frame receptions in *monitor* mode (Fig. 7.1 left). The monitoring station did not transmit and was not included in the TDMA round.

The experiments in real operational scenarios, taken during RoboCup games, include six robots, a base station and the monitoring station. The robots and base station were all integrated in the TDMA round. However, the base station was connected through Ethernet to the AP (Fig. 7.1 right).

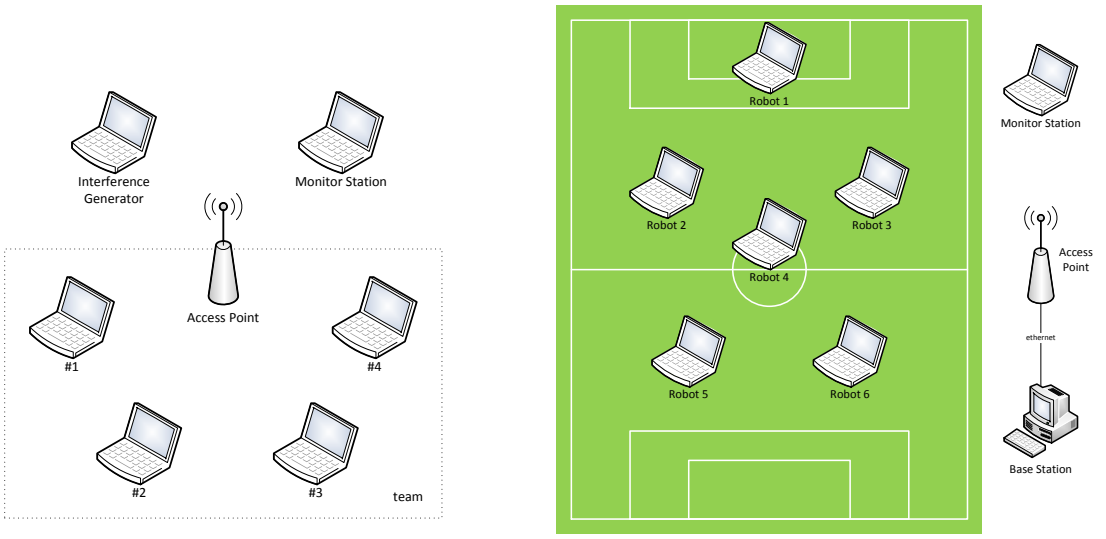


Figure 7.1: Laboratory (left) and game (right) setups

All the packets received in the configured frequency channel were saved in a file dump for later analysis. Note, however, that such logs do not necessarily represent the whole traffic in the network but just the perspective of the monitoring station. Nevertheless, despite using different computers in different experiments and different nodes, given that the physical layout of the experimental setup was relatively small in space, we believe the results shown are still representative of the traffic generally received by all nodes in the setup.

7.2 Comparing with no synchronization

These experiments aim at comparing the Reconfigurable and Adaptive TDMA protocol with a situation in which the same sources transmit the same information in the same way but without synchronization. In this case, the drifts of the respective nodes clocks can always lead to a situation in which several nodes will be transmitting approximately at the same time during a relevant interval as shown in Figure 5.3.

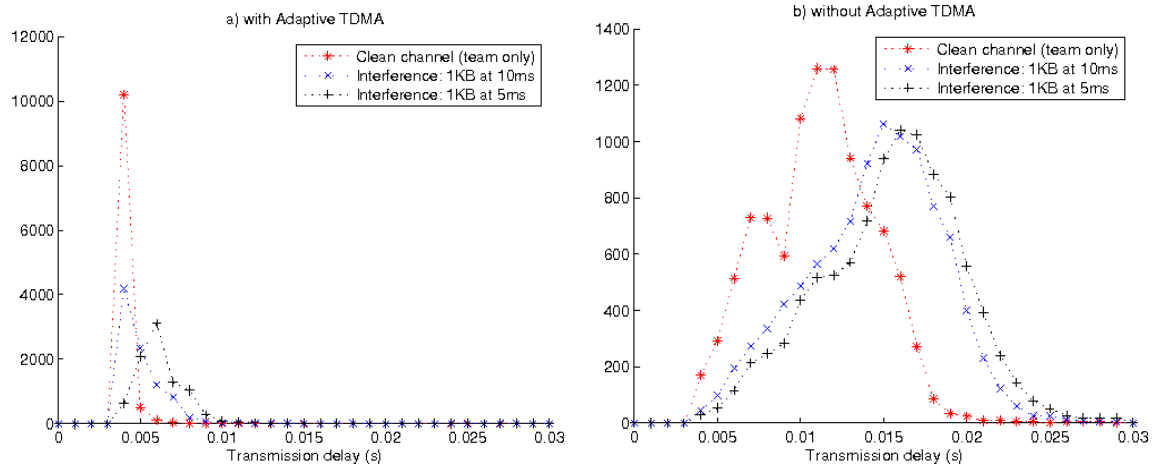
Therefore, we recreated the worst-case situation in which all robots will be transmitting at the same time causing maximum contention among the team members. This was achieved starting all nodes at the same time using a trigger signal sent by an external laptop used to generate interfering traffic. This was done in both cases, with and without Reconfigurable and Adaptive TDMA. We will see that, in the former case, the team communications will immediately be reorganized in the synchronized framework reducing collisions while in the latter they will continue colliding, generating a period of poor channel quality, independently of the channel load.

The experiments used IEEE 802.11b with multicast packets carrying 379 bytes of payload. The team included four nodes with a T_{tup} of $50ms$ and logs were extracted for about 9 minutes of continued operation. The interfering traffic was generated by an external laptop *pinging* the AP using $1000B$ packets at a rate of 5 and $10ms$ in two different experiments. The former case already corresponded to a saturated network. A third experiment was carried out without generating external traffic.

7.2.1 Latency measurements

The results concerning the wireless transmission delay are shown in the histograms of Fig. 7.2 with Reconfigurable and Adaptive TDMA (a) and without synchronization (b). In each case we include the three referred load situations.

It is clear that, in the former case, the synchronization imposed by the protocol immediately sorts out the high contention caused in the starting instant and the interference among team members is practically eliminated. In particular, from these measurements we can extract the T_{wt} parameter used in computing the age of the data items in the RTDB, as explained in Section 6.7. According to Fig. 7.2.a) we set $T_{wt} = 4ms$ with IEEE 802.11b. Based on similar experiments with IEEE 802.11a, in such case we use $T_{wt} = 1ms$.



a) with Reconfigurable and Adaptive TDMA

b) without synchronization

Figure 7.2: Transmission delay

Without synchronization, the team members continue interfering with each other, leading to a substantial increase in the transmission delay. Moreover, the impact of the interfering traffic is also worse without synchronization, showing the benefit of using our adaptive synchronized framework.

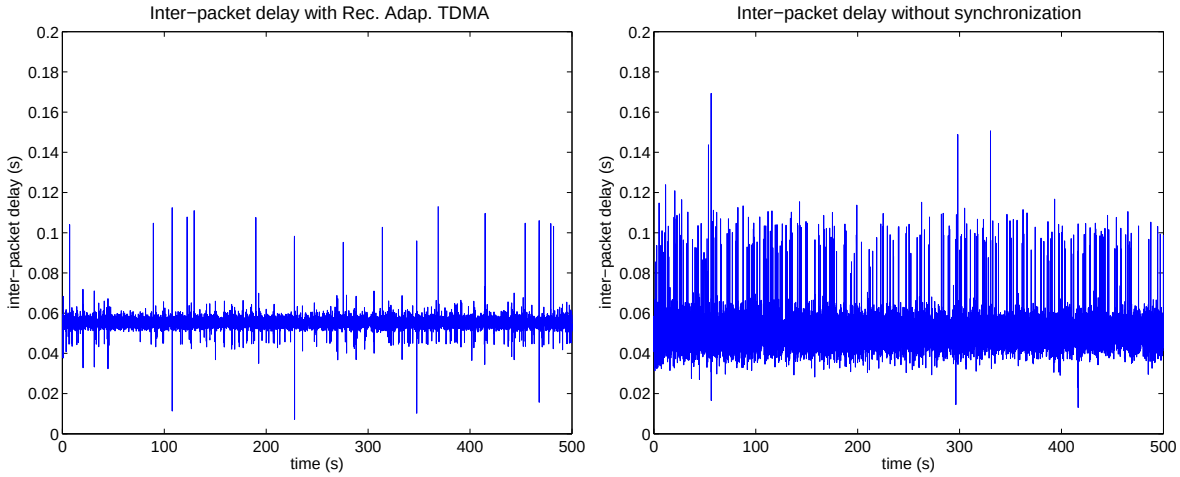
7.2.2 Packet losses

The impact in terms of packet losses is shown indirectly in Figure 7.3, which shows the intervals between consecutive packets received from another agent when using interfering `pings` every $10ms$. The other load cases presented similar patterns. Normal intervals are roughly between 50 and $60ms$ due to the adaptive feature of the protocol. Jumps to higher values represent consecutive losses, basically one loss per additional $50ms$ jump.

The case without synchronization shows much more continued losses given the high contention at the medium access and the high number of collisions.

Figure 7.4 shows the respective histograms of the number of consecutive lost packets for the three load cases, with and without synchronization. These histograms confirm the strong asymmetry between using and not using synchronization, with clearly higher figures in the latter case. They also show a surprising reduction in packet losses as the network load increases, without synchronization. The case without extra load is particularly visible.

This is paradoxal but illustrates a weakness of the IEEE 802.11 protocol. When two sources transmit at the same time, they both sense the medium free and trigger their communications immediately leading to an immediate collision. On the other hand, if there



a) with Reconfigurable and Adaptive TDMA

b) without synchronization

Figure 7.3: Timeline of the inter-packet interval from agent 0

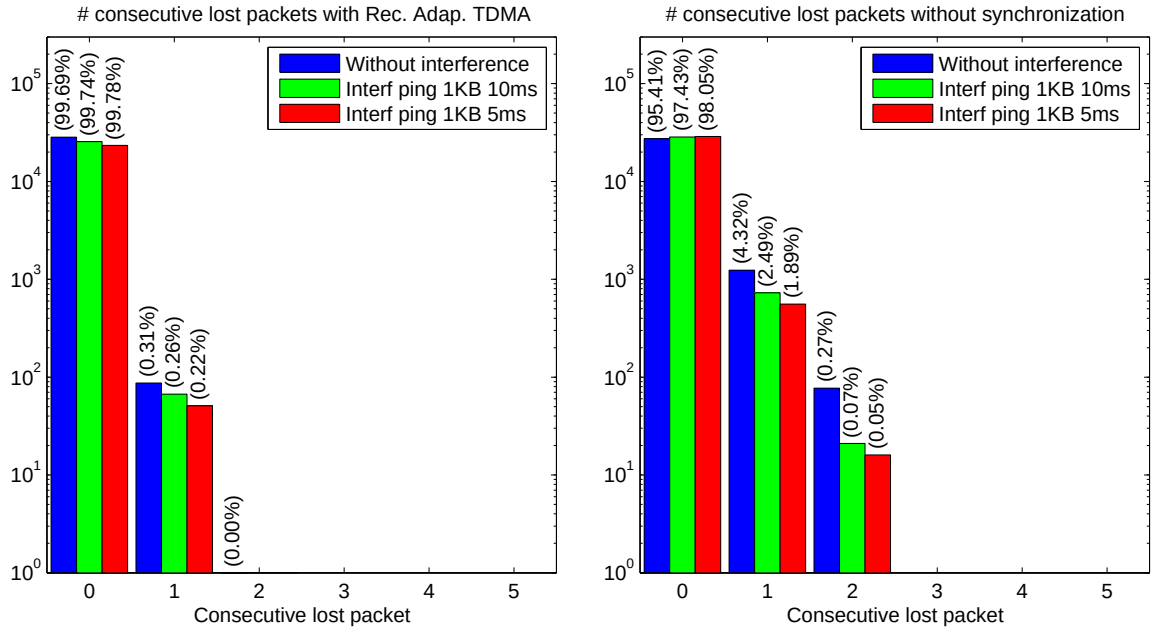
is more traffic in the network, then such sources will occasionally find the medium busy, leading to a backoff and retry, which is done according to the CSMA/CA rules that reduce collisions stochastically. As we increase the network load, the CSMA/CA arbitration rules will be applied more and more, reducing collisions and thus, packet losses.

These results show that it is beneficial using synchronization to improve the resilience of multicast packets and consequently, the timeliness of the system.

7.3 Comparing with non-adaptive TDMA

After having validated the positive impact of the TDMA kind of synchronization in reducing network delays and packet losses, it is also interesting to compare different alternatives to TDMA implementation, particularly with the typical one based on clock-synchronization.

When comparing Reconfigurable and Adaptive TDMA with clock synchronized TDMA there is one immediate difference. The former merges synchronization and data transmission while the latter needs a clock synchronization service and then implements data transmission alone. Moreover, the clock synchronization service requires transmissions of its own which must be accommodated by the data transmissions protocol. For these reasons we believe that Reconfigurable and Adaptive TDMA is simpler to deploy and use.



a) with Reconfigurable and Adaptive TDMA

b) without synchronization

Figure 7.4: Histograms of the number of consecutive lost packets

Another relevant issue is that clock synchronization algorithms typically use a master-slave approach, frequently without master redundancy, which makes them sensitive to single point failures. Conversely, Reconfigurable and Adaptive TDMA is fully distributed and thus resilient by nature to the failure of any of its nodes.

Finally, TDMA implementations based on clock synchronization are typically static. Thus, they perform poorly when facing coherent periodic external transmissions. In such cases, there will be recurrent periods of high interference that may cause significant degradation due to excessive collisions as shown in Figure 5.3. Conversely, the adaptive feature of Reconfigurable and Adaptive TDMA delays the round whenever the team transmissions suffer delays, quickly moving the phase of the TDMA round away from such interferences.

The following experiments aim at validating this feature of the protocol. Thus, we created growing levels of interference using the `ping` command from an external computer with large packets and varying burstiness to increase the impact of interference. The results were extracted from a sequence of logs of the team with four nodes, during $5min$ of operation, using a round of $99.5ms$. The reason for such period is to make it slightly different from that of the `ping` traffic, which was set to submultiples of $100ms$. With this difference in

periods, we expect a high contention interval between both types of traffic to occur recurrently approximately every $20s$, resulting from the difference of the frequencies of the interfering periodic processes.

The clock synchronized TDMA was implemented using the Chrony clock synchronization service, thus the respective results are referred as *chrony*. Those concerning the Reconfigurable and Adaptive TDMA are referred as *Rec Adap TDMA*.

7.3.1 Evolution of offsets and round period

Figure 7.5 shows the offsets of the nodes transmissions in the TDMA round, with respect to the transmissions of node 0, for both synchronization methods and with additional `ping` traffic of one $1KB$ packet every $20ms$. In this plot, we have excluded larger intervals caused by message losses for the sake of clarity. At the right side of the plot we show the mean value of the respective slot offset. The top line represents the interval between consecutive transmissions of node 0, thus showing the effective round period, which we refer to as $\tilde{T}_{tup}$.

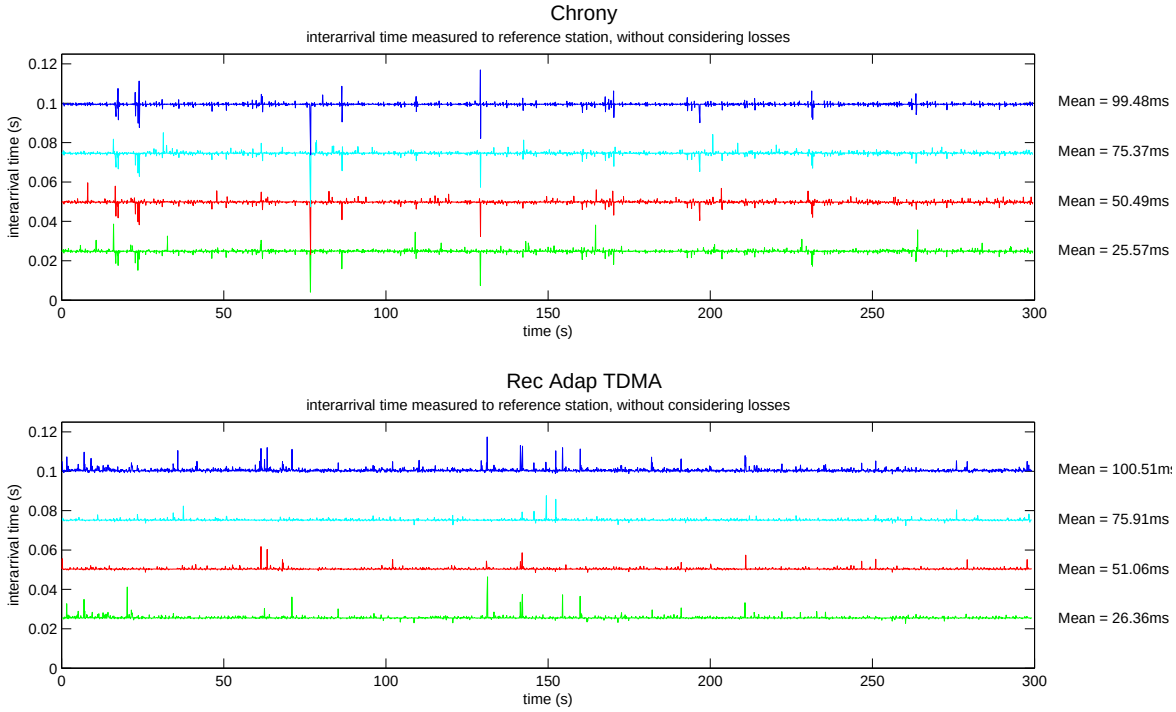


Figure 7.5: Clock synchronized TDMA versus Reconfigurable and Adaptive TDMA with $1KB$ `ping` every $20ms$ of external traffic

In general, we can observe a pattern in which the clock synchronized approach exhibits more symmetrical variations in the slot intervals, which is expected given the fixed regular average slot offsets. This is also observed with Reconfigurable and Adaptive TDMA for the slots, which use fixed offsets with respect to node 0 transmissions. However, the adaptive approach incorporates the delays suffered by the team transmissions in the round period, thus leading to effective periods that are typically longer than the programmed T_{tup}. This is seen in the top line of the plots. Conversely, the clock synchronized TDMA maintains an average round period close to the programmed one, i.e., $99.5ms$. Finally, the patterns were approximately similar for all other load cases.

The actual evolution of the measured round period for the Reconfigurable and Adaptive TDMA case is shown in Figure 7.6 as a function of the total average network load in each log. The figure shows a linear stochastic dependence, which is expected since more load generates more frequent delays in team packets, possibly longer, thus more frequent compensations.

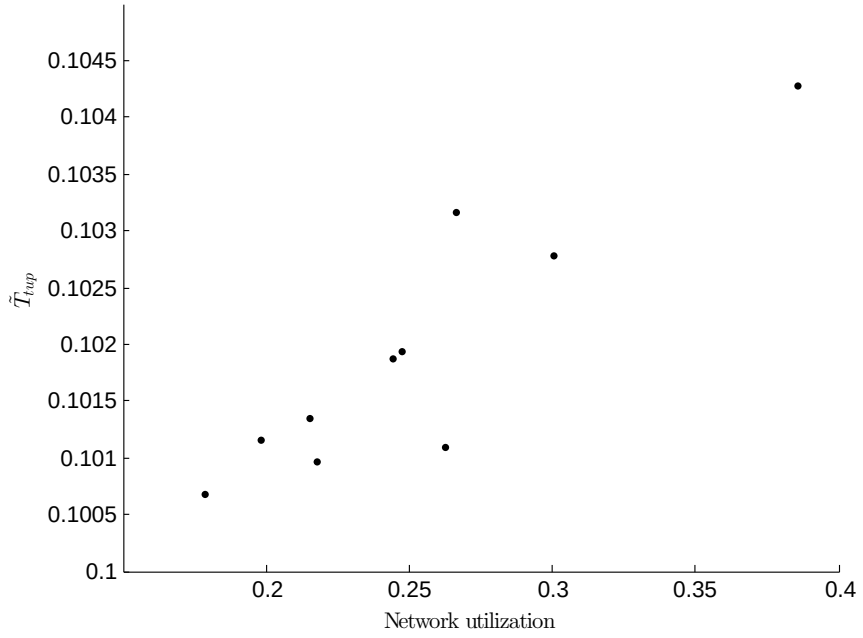


Figure 7.6: Effective round period ($\tilde{T}_{tup}$) of Reconfigurable and Adaptive TDMA for different total average loads

7.3.2 Packet losses with single packet interference

In order to compare the two synchronization methods in what concerns packet losses, we show the respective histograms in different situations. Fig. 7.7 shows such histograms for the cases without injected external `ping` traffic and with growing load of single $1KB$ packet `pings`. Note that, in the first case, there is still a certain residual level of external traffic that was circulating in the medium at the time the experiments were carried out.

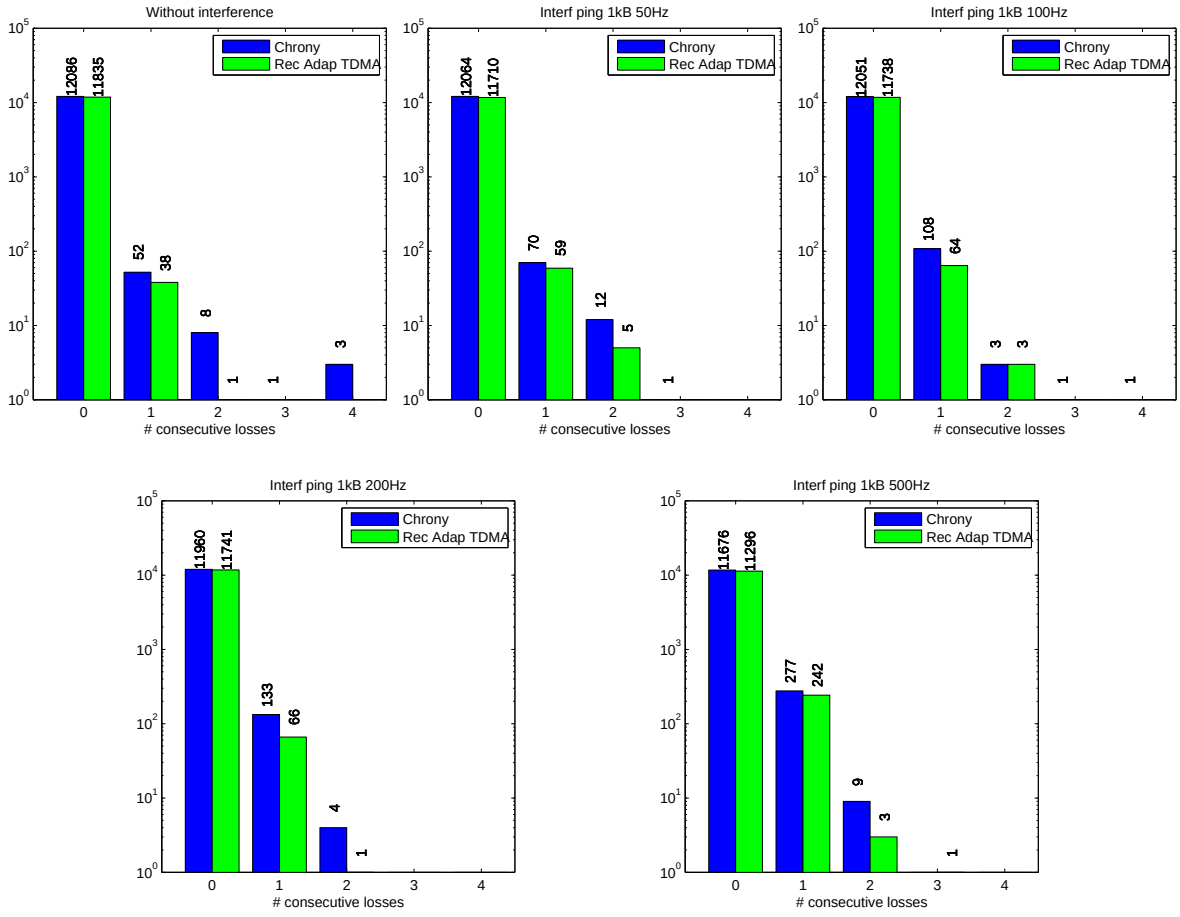


Figure 7.7: Histograms of consecutive lost packets with no `ping` , or $1KB$ single packet `ping` traffic with variable frequency

The results show an advantage of Reconfigurable and Adaptive TDMA which is expected. If the teams transmissions collide at a given moment with the interfering `ping` packets, eventually a team transmission will be delayed shifting the whole TDMA round so that the following transmissions move away from the interference.

This does not happen with the clock synchronized method, which keeps transmitting in moments of high collision probability until the clock drifts separate the transmitting instants of the interfering processes, which may take a significant amount of time.

7.3.3 Impact of external load bursts

We also tested the effect of external interference caused by bursty traffic. For this purpose we generated `ping` streams with $5KB$ and $10KB$, involving 4 and 7 consecutive MAC packets, respectively, most of them with $1500B$ payload.

The results are shown in the histograms of Figure 7.8. Interestingly, the results are different from the previous single interfering packet case. For lighter interfering loads, the performance in packets lost in both synchronization methods becomes approximately similar but for higher loads, the adaptive approach becomes worse.

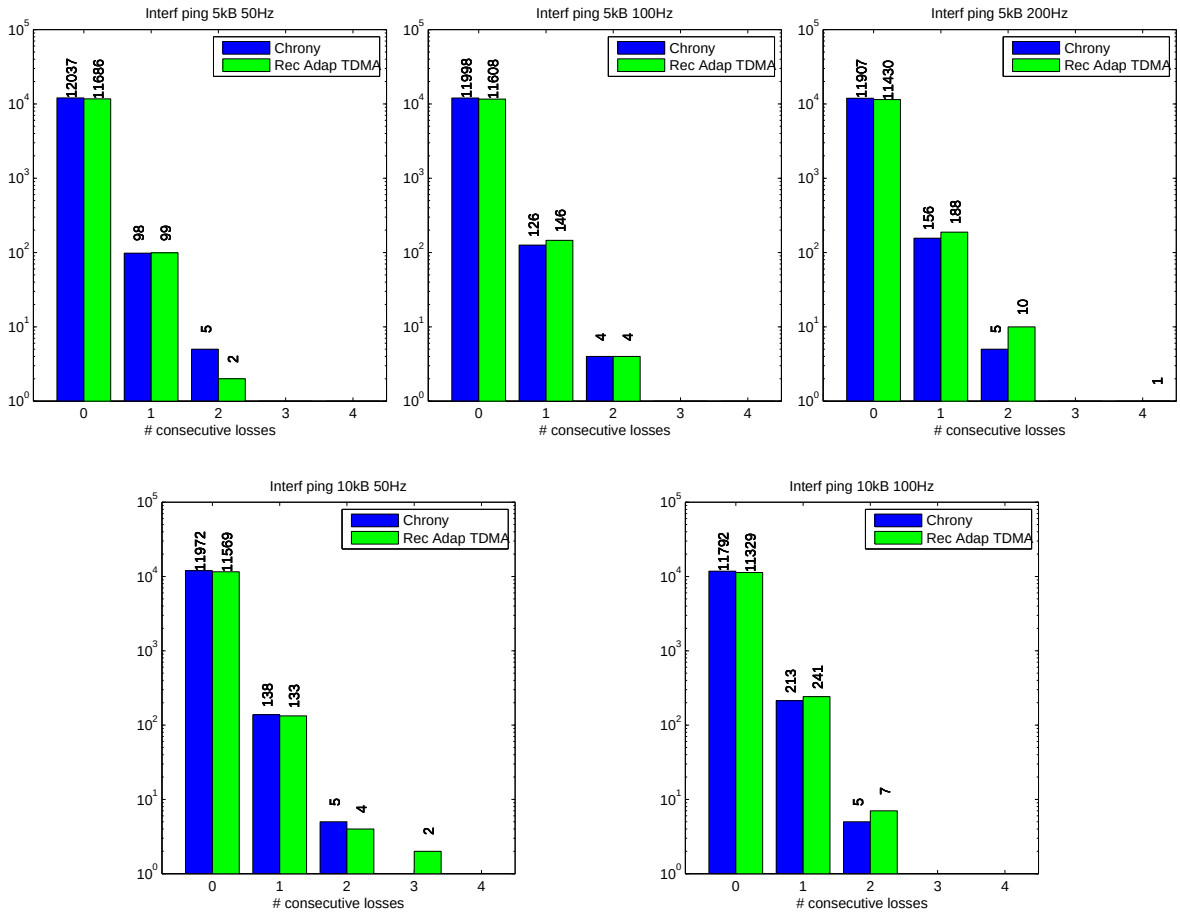


Figure 7.8: Histograms of consecutive lost packets with bursty `ping` traffic with 4 and 7 packet bursts and with variable frequency

We believe there are two different phenomena contributing to this situation. On one hand, bursty interferences have a higher probability of causing longer delays, in some cases beyond the adaptability threshold of the protocol (Δ_k as expressed in Eq. 5.12). Such delays will be ignored and the protocol will adapt less, tending to a behavior similar to that of clock synchronized TDMA.

The other phenomenon is related to the fact that the delays used by the adaptive method maybe be frequently shorter than the length of the bursts. This means that the protocol, trying to escape from an interference will still fall within the next instance of the interfering burst, a situation that can occur in several consecutive rounds. Note, too, that if the interfering stream has a slightly longer period than that of the team round, which is the case, delaying the team transmissions will most likely prolong the interference over an interval of more consecutive rounds.

In these cases, the clock synchronized TDMA approach will probably end up leaving the high contention period faster, thus leading to lower losses. However, for the general case of interference with non-bursty periodic interferences, the Reconfigurable and Adaptive TDMA approach is better, effectively avoiding periodic interferences and thus reducing collisions and packet losses.

Therefore, the Reconfigurable and Adaptive TDMA approach is particularly well suited when mitigating non-bursty interference. Nevertheless, its reconfigurable feature still makes it a better choice than the clock synchronized TDMA approach, in situations where the team composition is dynamic, maximizing the slots in each operational team configuration.

7.4 Operation in real scenarios

In this section we show logs obtained from a real operation scenario, namely from one RoboCup GermanOpen 2010 MSL game in Magdeburg, Germany, in which we used six mobile robots and a base station, thus seven nodes. We show the evolution of the team control information, namely the membership vector as perceived by each of the robots, as well as the intervals between consecutive transmissions for each of the robots. These logs show the correct protocol operation even in the presence of highly dynamic conditions, with frequent packet losses, including asymmetric ones, and robots that leave and join the team.

A plot of the slots offsets under dynamic team composition in this scenario is shown in Figure 7.9. This figure clearly shows the dynamic reconfiguration of the TDMA round according to the total number of running agents at each instant. It also illustrates the maximization of the time interval between consecutive transmissions of team members as the team composition changes.

Finally, we end the section with an analysis of the time that a team member that is outside the team takes to rejoin.

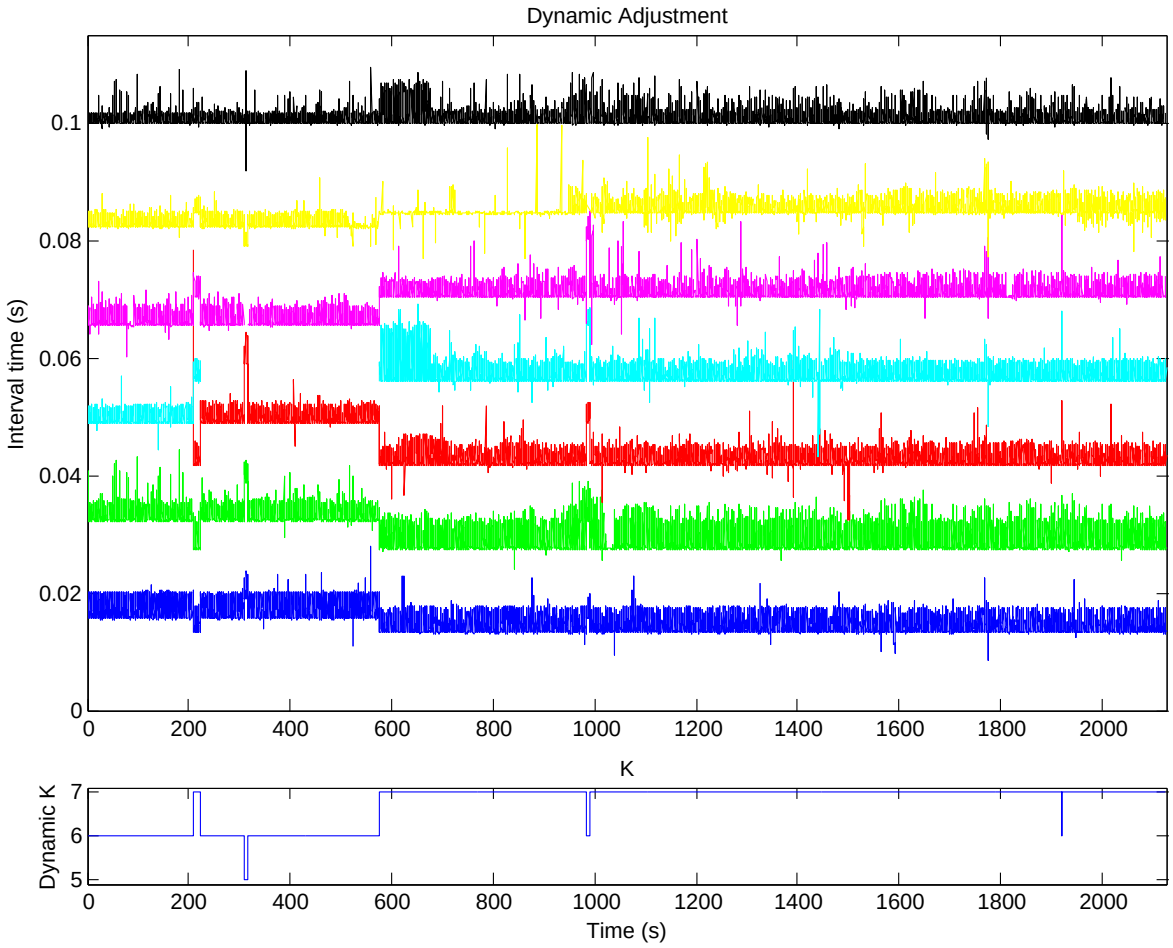


Figure 7.9: Evolution of round structure and number of nodes in the team

7.4.1 Membership vector evolution

As presented in Section 5.3, a membership vector is locally updated each round in each robot, keeping track of the receptions from other team members, and then broadcast to the others. Thus, all agents in the team keep their own membership vectors together with a copy of the membership vector of all other team mates. This allows the team to come up with a consistent view of the current team composition and thus configure the TDMA round appropriately.

Figures 7.10 to 7.12 show the dynamics of the team state, where we can see the evolution of the membership vector in each agent. The states of the robots, as perceived by each of them, are coded as:

0. Not Running
1. Insert
2. Running
3. Delete

By inspecting these logs we can see that agent 0 has better reception characteristics and typically receives from all active team members. This is expected as it corresponds to the base station, which has a cabled (Ethernet) connection to the AP. Thus, the communications with the base station are wireless just in the part between mobile agents and AP. All other agents, i.e., the mobile robots, show frequent packet losses, visible through the state oscillations between *2 – Running* and *3 – Delete*. However, such losses are filtered by the team management local state machines and do not generally cause variations in the team composition. Nevertheless, a few situations occur that quickly converge, as we explain next.

The actual team composition is the following. Initially, all agents except 3 are active. Agent 3 joins around second 200. Note that before being admitted, it sees all other nodes in state *1 – Insert*. Soon after, agent 4 leaves and rejoins shortly before second 600. These are the major changes clearly visible in the logs.

However, there are other short duration reconfigurations, typically caused by resets in mobile robots. For example, agent 5 leaves momentarily the team around second 300, being registered as state *0 – Not running* by the remaining agents, but rejoining soon after as indicated by the state *1 – Insert* in its own vector. The same happens with agent 6 slightly before second 1000 and after second 1900. Apart from these cases, there are occasional glitches caused by transient errors that do not cause any persisting inconsistency in the global team state.

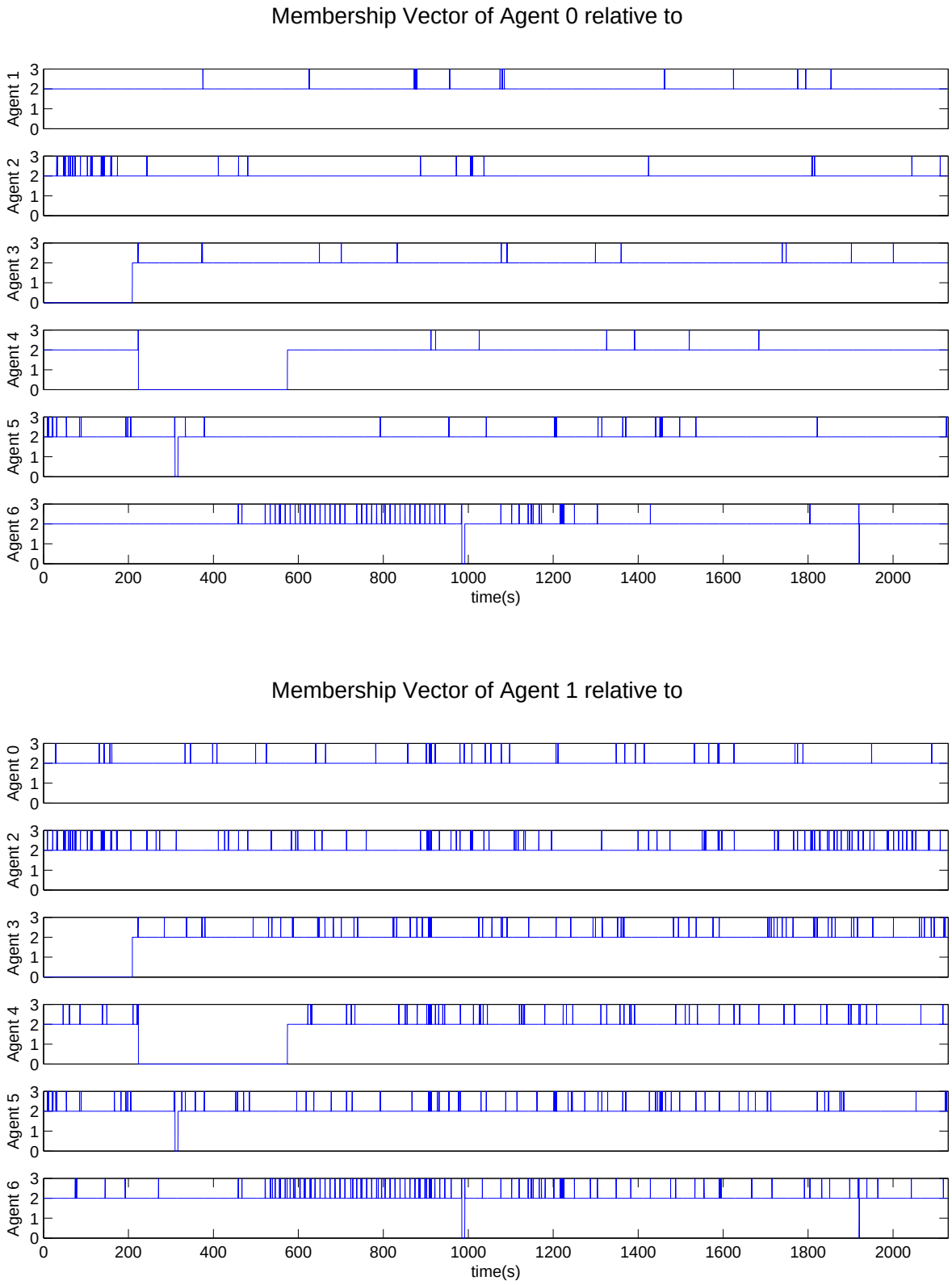


Figure 7.10: Membership Vector dynamics (1/3)

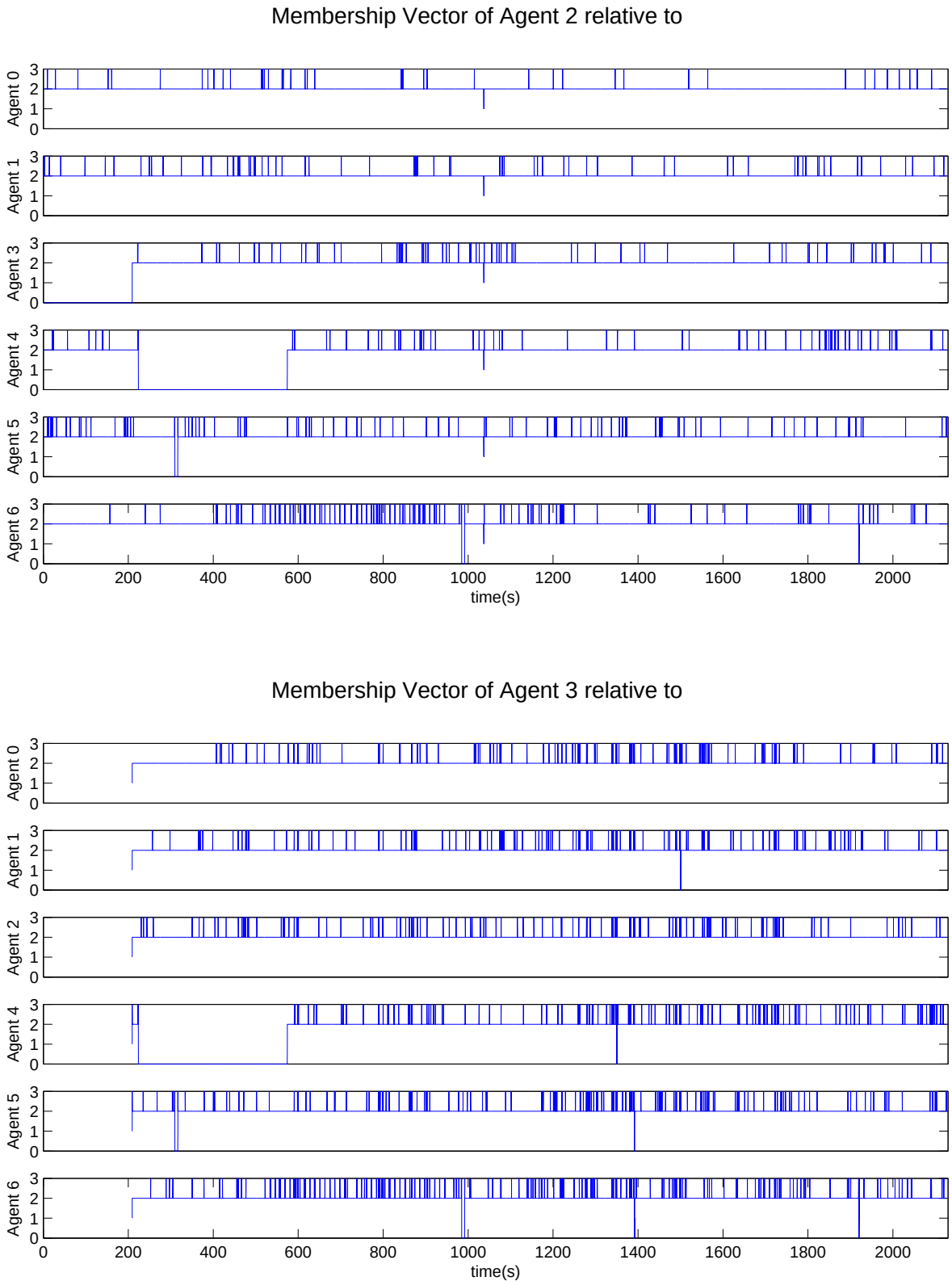


Figure 7.11: Membership Vector dynamics (2/3)

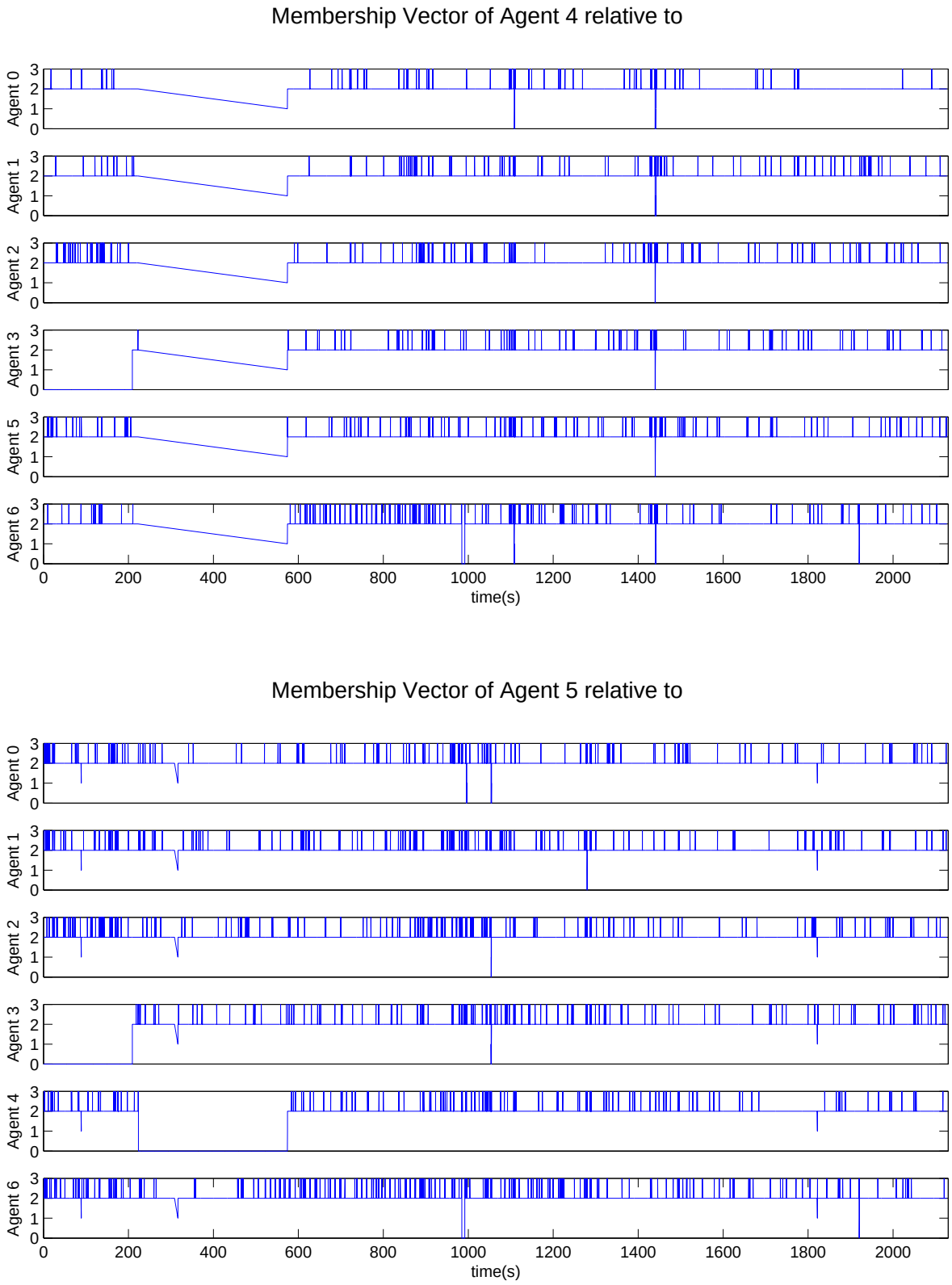


Figure 7.12: Membership Vector dynamics (3/3)

7.4.2 Intervals between consecutive transmissions

Figures 7.13 and 7.14 show the intervals between transmissions of the same agent, for agents 0 to 2 and 3 to 5, respectively. On the right we can see the associated histograms of consecutive packet losses. As expected, the adaptation mechanism forces baseline intervals that are close to but generally above the $100ms$ line. As we already explained before, when there is a packet loss we see a jump to $200ms$, and more consecutive losses imply additional $100ms$ of amplitude.

7.4.3 Time to join the team

Finally, we analyzed the time taken by team members that were outside the team to (re)join. This is a relevant parameter since until the join process is concluded, the node transmissions are unsynchronized with the team, thus with lower reliability.

Therefore, we measured the time to join of all joining processes detected in these logs, following the definition in Chapter 5, namely the interval A-D in Figure 5.14. This is the interval between the joining node first transmission and its first transmission in the new slot upon round reconfiguration. This interval should be upper bounded by Equation 5.14.

Figure 7.15 shows the measured joining intervals. As discussed in Section 5.3.3, this should vary between 1 and 2 rounds plus a fraction corresponding to the slot assigned to the new node. However, any error in a round during the joining phase will prolong this phase with one more round. Basically, the process ends with an error-free round, only.

The measured values are interestingly close to the minimum for each situation of extra rounds needed (represented by function $n(b, p)$ as in Section 5.3.4). This has a specific explanation. In fact, most joining cases were rejoins after a glitch departure, e.g., a reset of the network card while the communications process would continue working, attempting to transmit every round. Thus, when the network card became operational again, the communications process would continue transmitting very close to the slot that would be assigned upon reintegration, which was the slot that was assigned before. This is the case that leads to minimum joining time (Section 5.3.4).

In more general situations with agents that join at arbitrary times, joining times should exhibit a higher dispersion.

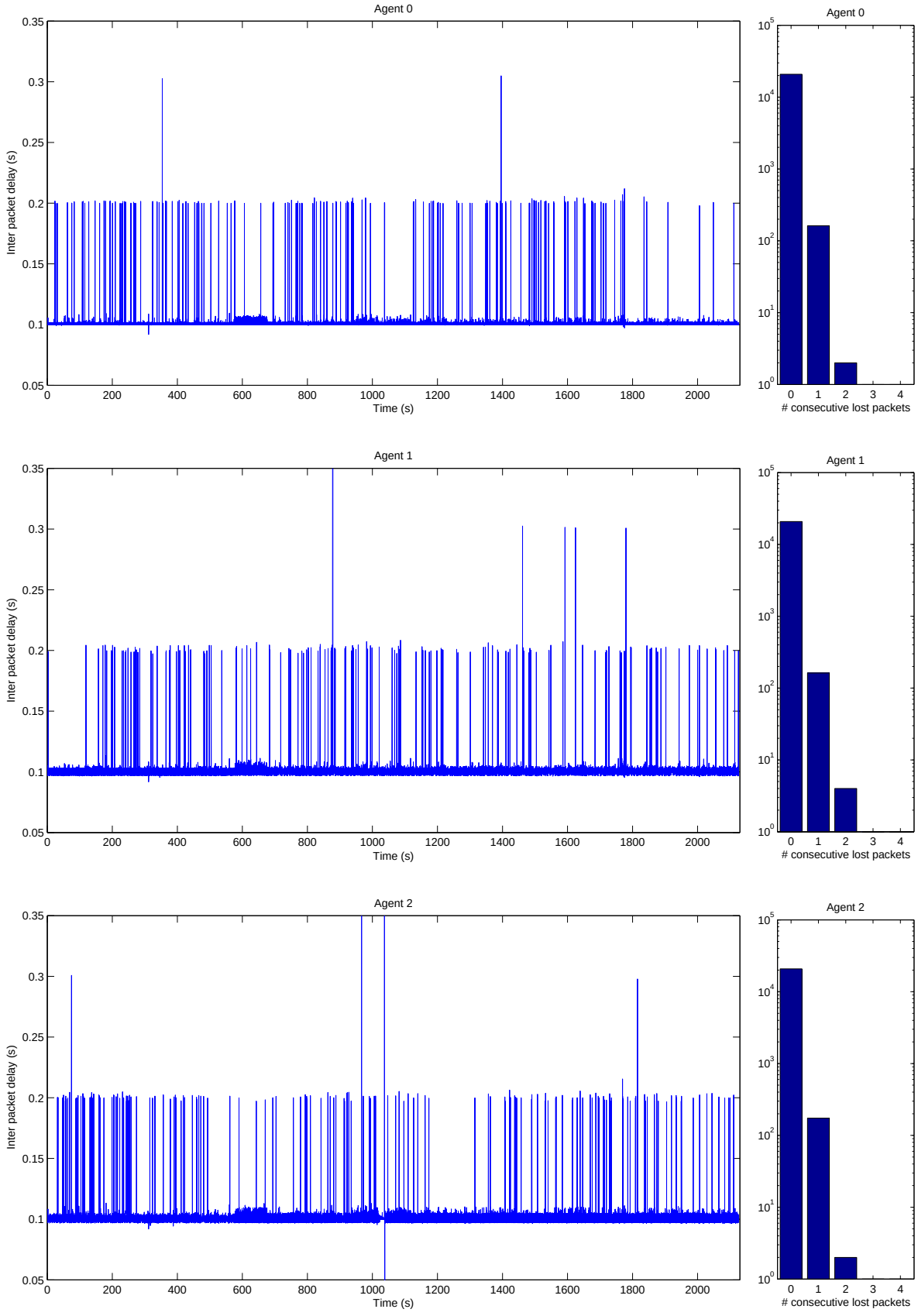


Figure 7.13: Timeline of the inter-packet delay (1/2)

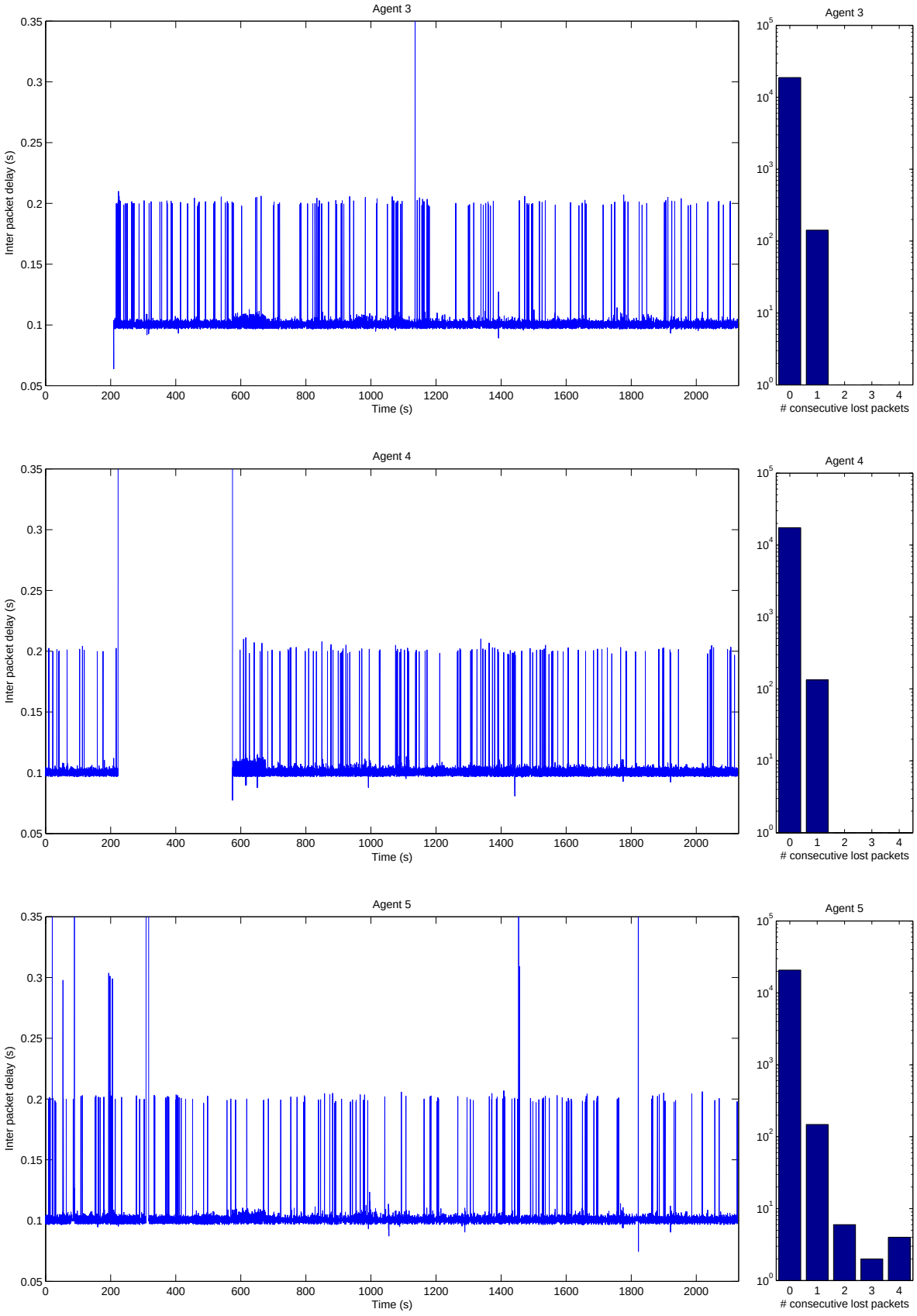


Figure 7.14: Timeline of the inter-packet delay (2/2)

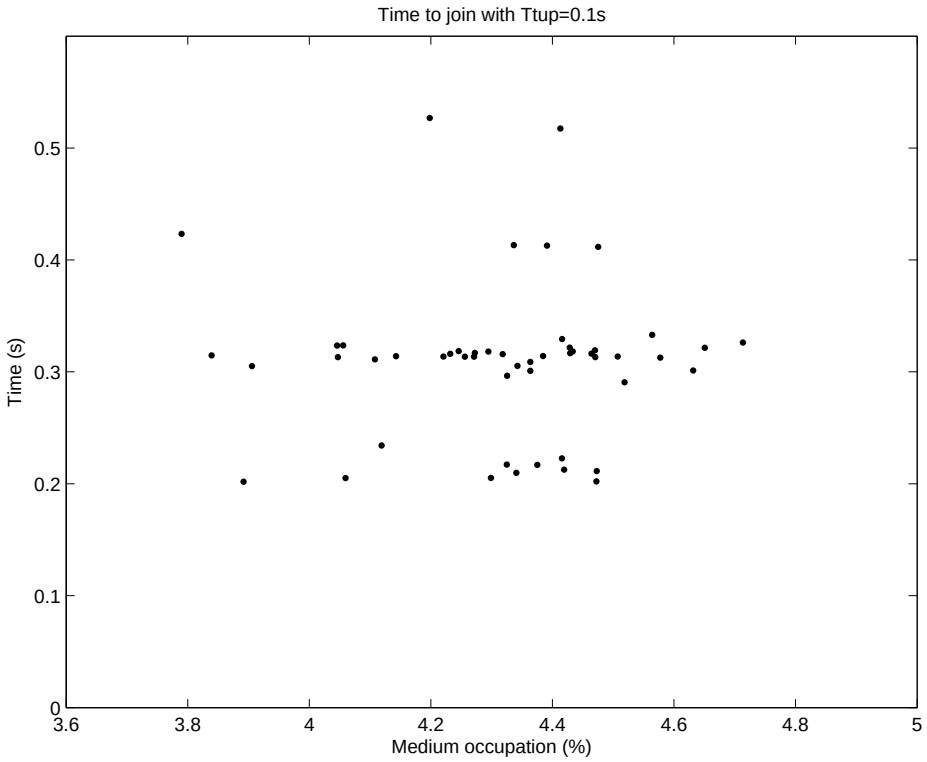


Figure 7.15: Time to join the team

In order to check the upper bound determined in Equation 5.14, we plotted the difference between this and measured values in Figure 7.16. The obtained values are all positive showing that they respect that upper bound. These results also give information on how far the upper bound is with respect to actually measured values. For the cases in this game, the upper bound is essentially between 1 and 2 rounds above the actual values.

Another information we extracted from these measurements is the distribution of extra rounds needed because of errors and collisions. This is given by function $n(b, p)$ where b is the bit-error rate and p the probability of successful packet reception. This distribution is shown in Figure 7.17 where we can see that most of the joining processes require none or one extra round, with just a few cases of more rounds needed.

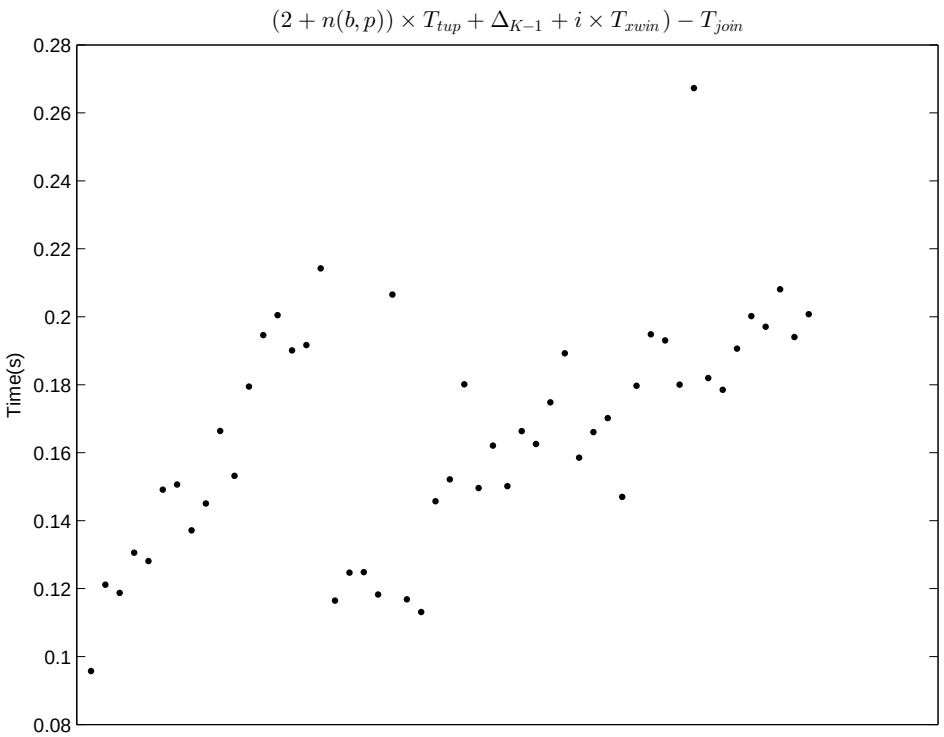


Figure 7.16: Difference between the upper bound and actual joining times

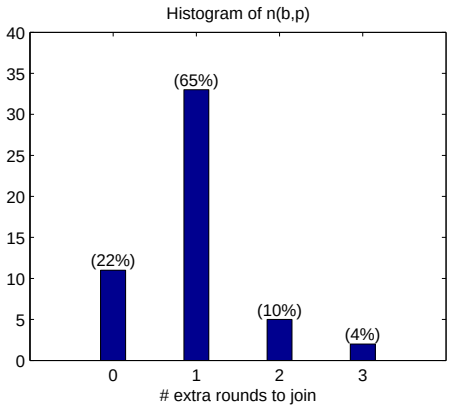


Figure 7.17: distribution of extra rounds needed in a joining process

7.5 Summary

In this chapter we showed several experimental results to validate the claimed properties of the proposed Reconfigurable and Adaptive TDMA protocol. Particularly, we showed that when comparing to non-synchronized periodic transmissions, our proposal reduces the transmission latency and the number of packet losses. Then we also compared to a clock

synchronized TDMA approach. We could see that we achieve better results in packet losses when the external load is not bursty, even for considerable loads. However, results with bursty interferences seem to indicate a better performance of the clock synchronized approach. Nevertheless, the reconfigurable feature of our protocol, that is capable of maximizing the slots assigned to the team agents in each instant, still outperforms the potential benefit of the clock synchronized approach with bursty periodic interferences.

Finally, we showed logs of a real operational scenario, in a RoboCup MSL game, which validated the team management mechanisms, in particular the global membership management which is used to drive the reconfiguration of the TDMA round. It was visible that even under highly dynamic conditions, the protocol maintained consistency and effectiveness in keeping a low number of packet losses. We ended this section with an analysis of the time to join the team actually taken by the joining agents in the game log. These measurements validated the analysis on the time to join presented in Section 5.3.4. We also used these experimental data to deduce the distribution of the $n(b, p)$ function, which gives us the number of extra rounds needed in the joining process, given a bit-error rate and a probability of successful packet reception.

Chapter 8

The CAMBADA RoboCup MSL team RTDB use case

The software of the CAMBADA robotic soccer team is built on top of the RTDB middleware with the Reconfigurable and Adaptive TDMA communication protocol behind. This chapter describes how the developed communication and database mechanisms support high-level cooperation and coordination in the team as well as the debug of team behaviors.

The RTDB middleware, together with the Reconfigurable and Adaptive TDMA protocol, is an open source project freely available at `http://code.google.com/p/rtdb/`. It was also adopted by other RoboCup MSL teams, such as Tech United, from the Technical University of Eindhoven [1], NuBot, from the National University of Defense Technology in China, and SocRob, from the Instituto Superior Técnico - Technical University of Lisbon [63]. The SPL Portuguese Team, from the Universities of Aveiro, Porto and Minho [70], a team of NAO humanoid robots competing in the RoboCup SPL, and CAMBADA@Home, from the University of Aveiro [7], with an autonomous robot for helping people with health problems and reduced capabilities, also use the developed middleware.

Beyond RoboCup, the RTDB was used in a national FCT-funded project, PCMMC - Perception-Driven Coordinated Multi-Robot Motion Control, conducted by the Technical University of Lisbon, University of Porto and Polytechnic Institute of Porto (PTDC/EEA-CRO/100692/2008)to support autonomous team formation control for improved global perception [73, 72].

Moreover, the Rota project with the Zinguer autonomous robot from the University of Aveiro competed in the Autonomous Driving League of the Portuguese Robotics Festival using the RTDB middleware for local inter-process communication, only, with the Reconfigurable and Adaptive TDMA used for debugging purposes.

These applications of our work allow us to qualitatively validate the second claim of our thesis, that states that the RTDB middleware facilitates the development of collaborative applications. We will elaborate more on the CAMBADA case study, since the RTDB was originally developed within and for CAMBADA.

8.1 The CAMBADA robotic soccer team

CAMBADA is the RoboCup MSL soccer team from the University of Aveiro. The project started in 2003, coordinated by the Transverse Activity on Intelligent Robotics of the Institute of Electronics and Telematics Engineering of Aveiro (IEETA). This project involves people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers, and the software development in areas such as image analysis and processing, sensor and information fusion, reasoning and control, cooperative sensing, communications among robots and the development of an efficient base station.

8.1.1 Hardware

The CAMBADA robots (Figure 8.1) were completely designed and built in-house. The base platform is cylindrical and the mechanical structure of the players is layered and modular. Each layer can be easily replaced by an equivalent one. The first layer contains the motors, wheels, batteries and an electromagnetic kicker. The second layer contains the control electronics and the third layer consists of a laptop computer, an omni-directional vision system, a frontal camera and an electronic compass. The players are capable of holonomic motion using three omni-directional roller wheels.

The architecture of the robots follows a biomorphic paradigm (see Fig. 8.2), being centered on a main computer (a laptop), *the brain*, which is responsible for the higher-level behavior coordination, i.e. the coordination layer. This main computer also handles external communication with the other robots as well as high bandwidth sensors, typically vision, directly attached to it.

Finally, this unit receives low bandwidth sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system (Figure 8.3), *the nervous system*, that uses a Controller Area Network (CAN) bus.

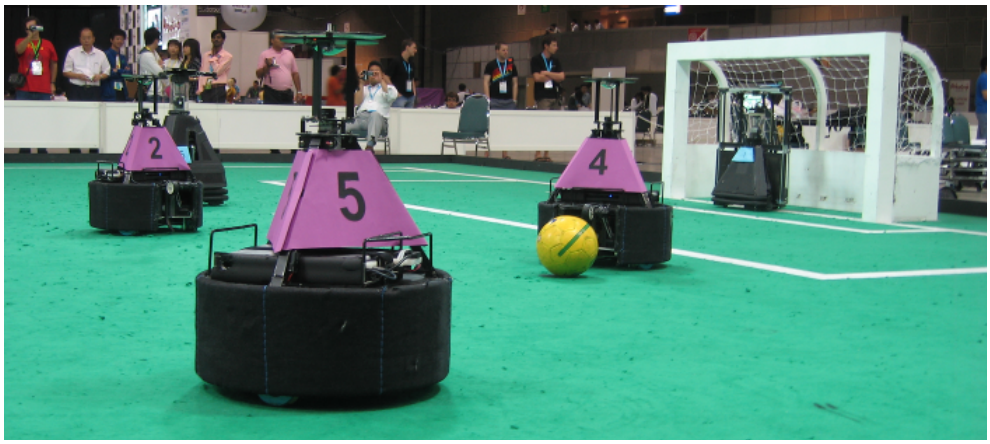


Figure 8.1: Robots used by the CAMBADA MSL robotic soccer team

8.1.2 Software

The software architecture is also distributed. In each robot, five different processes execute concurrently in the laptop computer with Linux. All the processes run at the robot's processing unit in Linux and communicate using the RTDB (Annex B). This architecture is shown in Figure 8.4.

The main processes running in the CAMBADA robots are the following:

Vision is responsible for acquiring the visual data from the vision system cameras, processing the data to extract relevant features and transmitting the relevant information to the CAMBADA agent namely the position of the ball, the lines detected for localization purposes and positions of obstacles. Given the structured environment in which the robots play, all this data is currently acquired by color segmentation [69, 71] but it is correlated with shape, e.g., the ball detection.

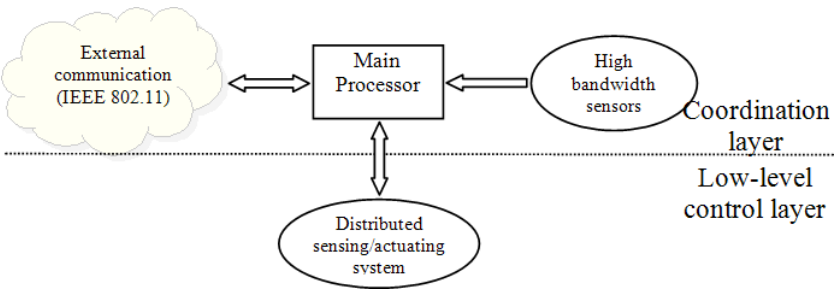


Figure 8.2: The biomorphic architecture of the CAMBADA robots

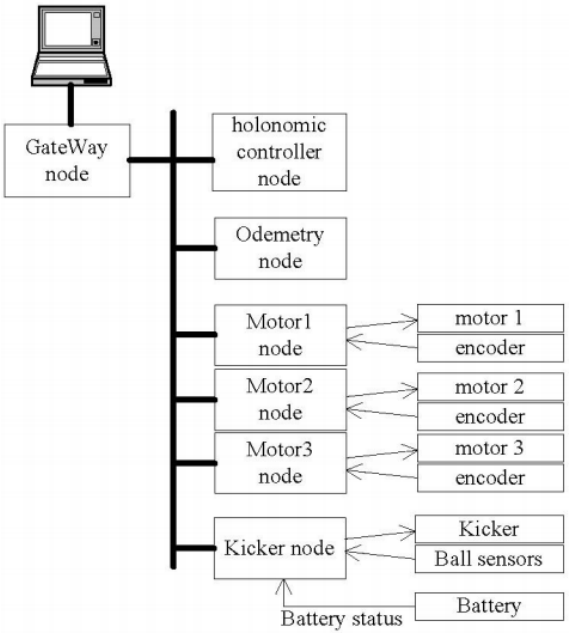


Figure 8.3: Hardware architecture with functional mapping [92]

Agent is the process that integrates sensor information and constructs the robot's world state. The agent then decides the command to be applied, based on the perception of the worldstate, according to a predefined strategy [51, 52].

Comm handles the inter-robot communication, receiving the information shared by the team mates and transmitting the data from the shared section of its own RTDB [86, 87].

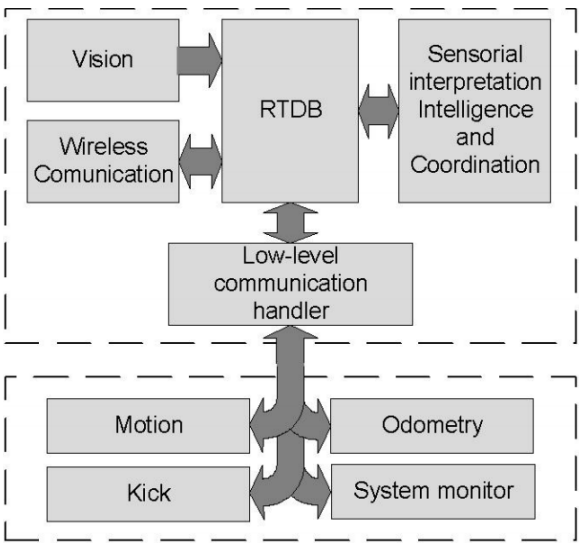


Figure 8.4: Layered software architecture of CAMBADA players [8]

HWcomm is the hardware communication process responsible for transmitting the data to and receiving data from the low-level sensing and actuation system.

Monitor checks the state of the remaining processes relaunching them in case of abnormal termination.

8.2 Building cooperative behaviors on top of the RTDB

The complete control software of the CAMBADA robots is relatively complex and includes several behaviors designed for specific phases of the game. In this section we refer to two of them as illustrative examples that use the RTDB.

8.2.1 Collaborative ball detection

Figure 8.5 illustrates the general ball detection and tracking approach adopted in the CAMBADA team.

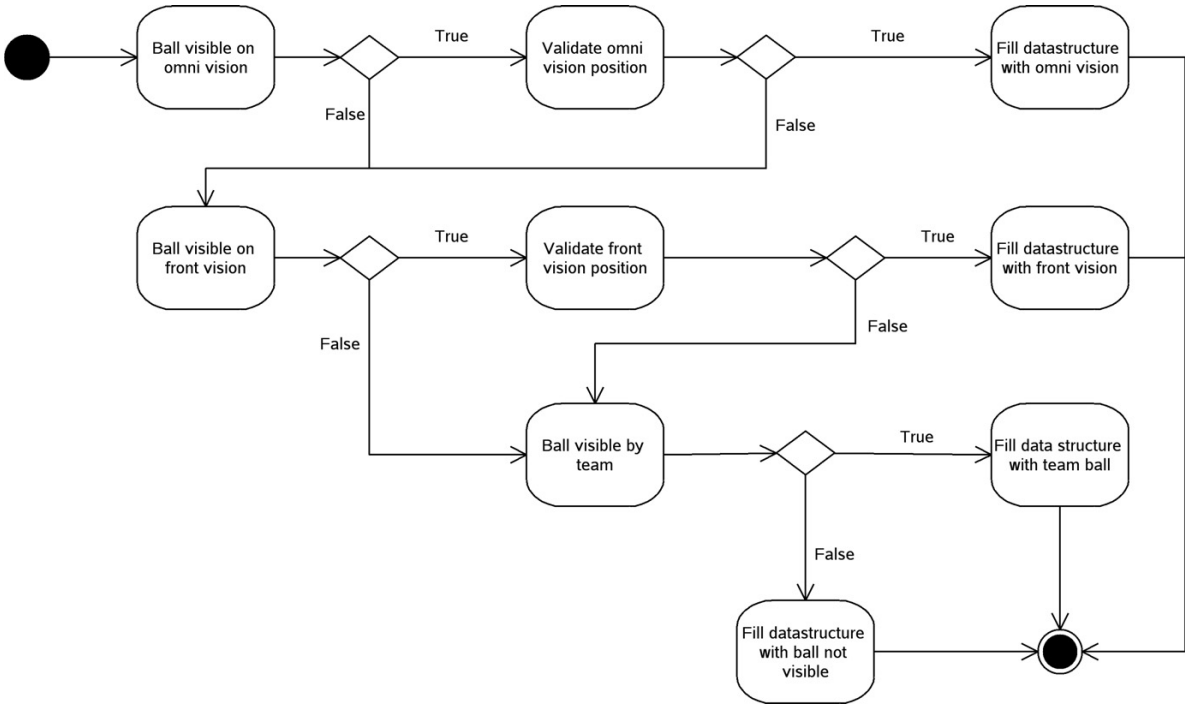


Figure 8.5: Collaborative ball detection behavior [68]

Due to the central role of the ball in a soccer game, when a robot cannot detect it by its own visual sensors (omni or frontal camera), it is still important to try to know the position of the ball by consulting with the team mates. This is achieved by having each robot sharing its perception of the ball position in the RTDB.

Using the age field of the ball data structure in the RTDB, it is possible to know for how many cycles its position was not updated. When the age of the last seen ball position is more than a given number of cycles the robot assumes that it cannot detect the ball on its own at that moment.

When the ball is not visible, or its perceived position is not valid, the robot uses the information of the ball position communicated by the other running team mates to know where the ball is. Note that this information is available locally in the RTDB and can be accessed without using communications. The ball positions as detected by the other robots in the team are averaged and the standard deviation computed. This is done to define a validity region around the average. Outliers are considered invalid positions. Then, the valid ball information of the team mate that has a shorter distance to the ball is chosen [51].

When detecting the ball directly, the same algorithm is used to validate the information, eliminating possible fake balls detection.

8.2.2 Strategy and coordination

In CAMBADA, each robot is an autonomous agent that coordinates its actions with its team mates through communication and information exchange over the RTDB. In order to achieve effective cooperation, each individual robot must be integrated in a global team strategy that defines roles and behaviors, the latter being basic robot sensorimotor skills like moving to a given location, and the former being selections of specific behaviors to be applied at each instant.

In general, coordination techniques have long been explored in the RoboCup Soccer Simulation League (SSL) [84, 96] and soon after applied to the MSL [100] upon a few necessary modifications. One of such techniques is formation control in which each player receives a coordinated motion behavior. In the case of the CAMBADA team [51, 52], the agents can be dynamically assigned to specific positionings. A positioning is defined in a strategy configuration file based on three items of information:

Home position is the assigned starting player position, used when (re)starting a game;

Region is the part of the field where the player can move;

Ball attraction parameters used to compute how a player is attracted to the ball.

Different home positions and ball attraction parameters lead to different strategic movement models, from defensive to wing, midfielder and attack. Figure 8.6 shows an example of team formation for several ball positions.

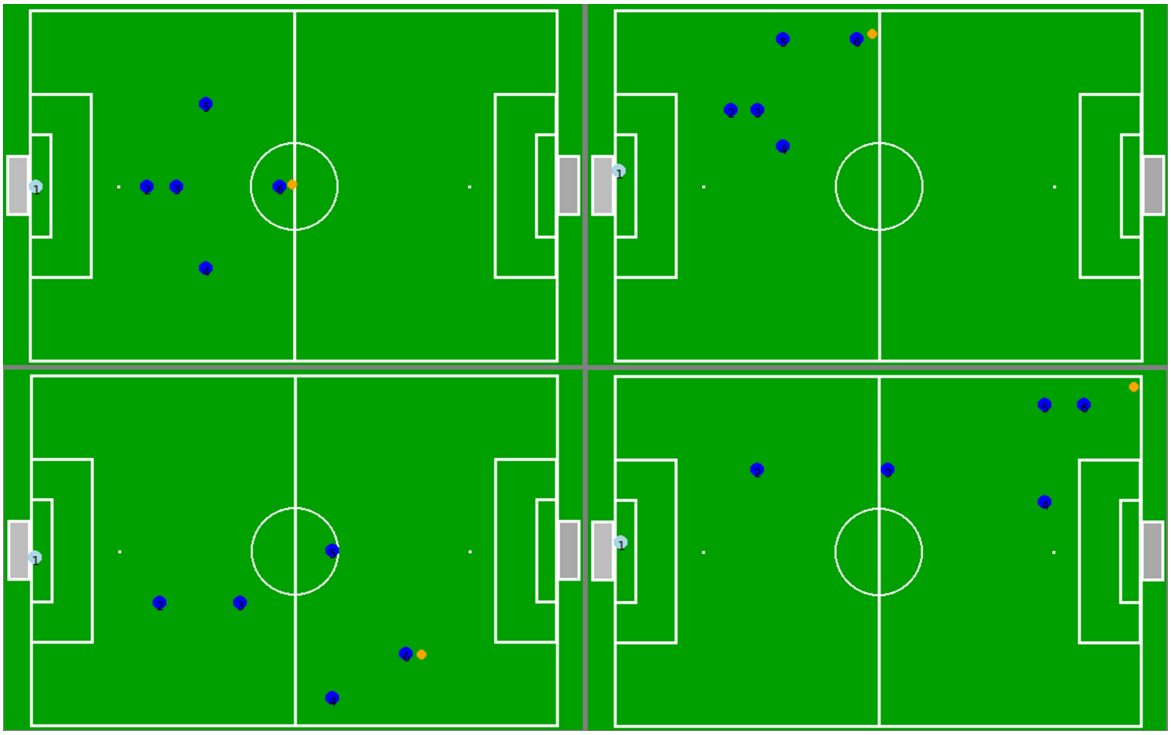


Figure 8.6: CAMBADA team coordination in some different game situations [50]

The team coach, running in the base station, is able of adapting the team strategy according to the current game situation. For example, if the team is not scoring, it might be important to define more robots as attackers. Or in an important game where the team should not suffer any goal, it might be better to increase the number of defenders.

The parameters that guide the strategy are shared through the RTDB (see Annex B) with the player robots, defining their roles and ultimately their behavior in the game. The state sharing nature of the RTDB is very adequate to support this kind of applications since the robots receive continuously a refresh of their current roles avoiding, for example, the possibility of missing a role change and continue on a wrong role. Moreover, the RTDB facilitates the synchronized roles reassignment, which enhances team coordinated movements, and particularly dynamic formations.

8.3 Debugging high level behaviors with the RTDB

When programming collaborative behaviors, debugging becomes more complex because it involves the state of all the robots in the team. Accessing all such states simultaneously is a problem since they are local. One typical approach is to have them logged locally and analyzed jointly later on, offline. This is rather inefficient, though, imposing long test-correction cycles and, moreover, it requires clock synchronization in order to allow recovering a consistent vision of the global team.

Alternatively, these difficulties can be significantly reduced using the RTDB. In this case, the RTDB at each agent is configured to share relevant local state variables, just during the debug phase. This simple action allows receiving all such states, inherently synchronized, in a base station for online analysis [28]. Moreover, the monitoring station can also log these states consistently, which allows them to be replayed back for offline analysis, if desired, at any time without losing the synchronization.

Beyond the mere observation of distributed state, the RTDB can also support interactive tuning of such collaborative behaviors by including in the base station configuration parameters that are shared online with the agents and used by these to adjust their behaviors locally.

In the specific case of the CAMBADA team, several commonly relevant state variables are always shared by the robots and displayed in a base station, such as their positions. The base station also receives the ball position as detected by each robot and fuses all such individual perspectives in a single consistent ball position that is displayed on the screen, in the game field area (Figure 8.7).

There are also other state variables that are shared, such as role and behavior state variables, as well as an operational vector that indicates, for each of the main subsystems of the robot, the run/stop state, battery level and health status. This information is displayed in the lower part of the screen. On the right side of the screen there is an area that displays game state information as received from the referee box that is used by the referees.

This same base station interface allows sending basic commands to the team, such as global and individual start/stop, go to home positions, return to the team support area, etc.

Beyond these general parameters that are exchanged between the players and the base station, there is also a debugging variable per robot that allows receiving in the base station any additional internal robot variable that the programmer wishes to track online. This debugging variable is a two dimensions vector that can represent a position on the game field, such as the locally fused ball position that the robots use internally, or the position of

Figure 8.7: The base station *Main Window* [28]

the opponent goal as seen by each robot, etc. Then, there is a switch in the base station that, if set, allows displaying the debug point in the game field online, together with the representation of the robots.

8.4 Summary

In this chapter we showed some particular aspects of the main use case of the RTDB middleware, which is the RoboCup MSL team CAMBADA, from the University of Aveiro. We briefly described this team and then showed how the RTDB is effectively used to support collaborative behaviors with a couple of examples, namely the collaborative ball tracking and the global team strategy execution.

Then we discussed how the RTDB can be used to effectively solve the problem of distributed state observation and logging, as well as to support the debugging of high level global team behaviors.

We believe that the continued use of the RTDB by the CAMBADA team, which was then adopted by other MSL teams and several other robotics projects, validates the second thesis claim that it is an effective platform to develop collaborative behaviors, simplifying the respective development, deployment and control. In fact, this adoption and use of our middleware is an indirect confirmation of its advantage with respect to the other existing middleware alternatives.

Chapter 9

Conclusions

Cooperative robotics has the potential to increase the performance and robustness in many applications domains. One example is area coverage applications, such as search and rescue, surveillance or cleaning. Another example is robotic applications where robots needs to operate continuously. In this case, faulty robots can be immediately replaced by back up robots. Yet another example is robotic applications where expensive actuating robots can be complemented with multiple inexpensive sensing robots. This way, cooperative robotics has become a topic of high interest in the robotics and multi-agent systems research communities.

Developing, deploying and operating such teams of cooperating robots raises many difficulties associated with the needed synchronization and information sharing, many of which are related with the use of a wireless communication medium. Therefore, making a rational, parsimonious and organized utilization of the communication channel becomes particularly relevant, which is related with the control of the transmissions but also with the organization and storage of the data.

The work we carried out in this Thesis, which fits in this scope, produced tools to support the development and operation of teams of cooperating autonomous robots. In particular, it was motivated and essentially developed within RoboCup, which is one example of an initiative aiming at fostering research in cooperative robotics through competitions involving teams of robots. Within this initiative, which includes many different competitions, we focused on the robotic soccer Middle-Size League where all robots of each team are fully autonomous and global knowledge is unavailable.

Nevertheless, in spite of the RoboCup motivation, the final results are usable well beyond that initiative and have, in fact, been successfully used in different scopes. In this chapter we briefly revisit our research contributions and we discuss open research lines for future work.

9.1 Revisiting the contributions

This work led to two main contributions, namely the Reconfigurable and Adaptive TDMA protocol, explained and analyzed in detail in Chapter 5, and the RTDB light middleware, presented and discussed in Chapter 6.

The former is a specific Wi-Fi-based wireless communication protocol that reduces medium access collisions among team members in the presence of uncontrolled interfering (allien) traffic, using a modified TDMA scheme. We named it Reconfigurable and Adaptive TDMA highlighting its capability to automatically reconfigure the TDMA round structure to the actual number of active team members at each instant, as well as the capability to adjust the round phase to escape from persistent periodic interfering traffic. These features contribute to a better timeliness of the communications.

The Reconfigurable and Adaptive TDMA protocol sets a cyclic framework that allows maintaining all robots in the team synchronized to each other without resorting to clock synchronization.

The latter contribution is a novel middleware to support collaborative behaviors that relies on a real-time database, partially replicated, containing both local and remote state variables, in a distributed shared memory style.

The RTDB creates a temporal gateway between the internal processes in each robot and the communication activity. In fact, user processes have access to remote variables through local copies (proxies), while the communication activity to keep these proxies updated is carried out transparently in the background at an adequate rate that ensures their temporal validity. Therefore, the RTDB access primitives are local, not including the communication delays, thus contributing to a better timeliness of the local processes execution.

An important real-time feature of the RTDB is the provision of age information together with every RTDB access to remote data. This is achieved with a novel time-stamping mechanism that lets the application know how old a data item is at the time of consumption without a global synchronized clock. With this knowledge, applications can detect stale data as well as estimate the evolution of each item using appropriate temporal models.

A last feature that we would like to highlight in this summary for its importance in the development of real-time distributed applications arising from a combination of the RTDB with the synchronous Reconfigurable and Adaptive TDMA protocol. In fact, this combination enables the remote observation of local variables in a synchronized way, as well as their consistent logging. This feature also supports the consistent log replay of the whole team state, a very important feature to analyze and correct collaborative behaviors.

Finally, this work also generated a third contribution specifically within RoboCup, which is a study and characterization of the communications in the Middle Size League, which was presented in Chapter 4.

9.2 Validating the thesis

The Thesis that we presented in Chapter 1 contains two main claims. The first one claims that using a self-organizing approach in the communications that take place within a team of cooperating agents reduces mutual interferences and thus contributes to reduce the network delay and packet losses. Particularly, we claim that the proposed Reconfigurable and Adaptive TDMA protocol achieves such benefits without needing additional traffic control mechanisms or clock synchronization.

The validation of this claim is essentially carried out through experiments, which are shown in Chapter 7. We show that the Reconfigurable and Adaptive TDMA protocol achieves lower network delays and packet losses under severe mutual interference than an approach in which the team members transmit periodically but without synchronization. On the other hand, we show that the adaptive feature of the protocol allows escaping from persistent periodic interfering traffic with coherent periods, while a traditional clock synchronized TDMA approach does not. However, we observed that such advantage disappears when the interfering traffic is strongly bursty. Nevertheless, the clock synchronized approach also lacks the reconfigurable capability of the protocol, leading to lower capacity to accommodate interfering traffic when just a small number of the team agents are active.

Finally, we validated the operation of the Reconfigurable and Adaptive TDMA protocol in many real operational scenarios and we discussed logs of one specific RoboCup MSL game that illustrates the protocol dynamics.

The second claim states that the distributed shared memory paradigm provides an adequate abstraction to facilitate the development, deployment and operation of cooperative behaviors. In particular, we claim that the RTDB middleware, following such paradigm, provides such benefits while exhibiting a parsimonious use of the network and enhanced age information when accessing remote variables.

This claim was essentially validated in a qualitative way in Chapter 8 by showing a non-trivial use case in which the RTDB middleware was clearly beneficial. Moreover, we believe that the positive impact and receptivity that the RTDB middleware had in the RoboCup MSL community, as well as in other RoboCup leagues and in several other robotic applications also contribute to validate our claim.

9.3 Future work

The work conducted in the context of this thesis unveiled some interesting research ideas that are worth further exploration. In this section we refer to several of these ideas, some of which are more practical and others more fundamental.

Ad-hoc topology

One research line that evolved from this work concerned the extension to an ad-hoc topology. In this case, there is no AP and the information to share among the team members needs to be propagated through the network, for example, using flooding. The synchronization among the nodes becomes more complex to enforce so as consistency across the team. This line has been researched in parallel with our thesis work along the past years and some results can be observed in [73, 72].

Our research has, however, stayed with infrastructured topologies where the presence of an AP helps significantly in enforcing consistency across the team even under highly dynamic scenarios. We believe that the use of an AP does not necessarily impose a strong limitation in the range of possible applications. In fact, our robotic team can be deployed in unstructured areas and the AP can be carried by one of the team members that can position itself in a way to maximize the team coverage. Alternatively, the AP can stay near a monitoring station, positioned in a convenient location in the operational area.

The following are open research lines that apply to our AP-based approach.

Generic traffic interface

Our Reconfigurable and Adaptive TDMA protocol has been designed with a specific interface to support the RTDB, allowing only a few Wi-Fi packets transmitted in each slot. This is a limitation since there may be situations in which the protocol can be useful without the RTDB on one hand, and could be applied in an ordinary protocol stack in a general purpose operating system on the other, to support generic communications from any communicating application. The interest in this possibility arises from the capacity of our protocol to organize the communications among a set of transmitters reducing collisions under high traffic loads, e.g., several servers streaming video to diverse clients.

The insertion of our protocol in a protocol stack, namely between the Wi-Fi driver and the IP stack, seems simple to achieve using either `iptables` or Tun/Tap interfaces. The challenge consists in controlling the amount of traffic that a node can transmit during its slot to make the best possible use of its duration while still avoiding overruns that would destroy the mutual isolation between slots.

This line has already been started but there are no clear results, yet.

Dynamic adjustment of system parameters

Despite the adaptive and reconfigurable features of our protocol, there are a few parameters defined statically, by configuration, such as the TDMA round (T_{tup}), which establishes the responsiveness and temporal resolution of the protocol, and ϵ, which defines the range of delays to which the protocol will adapt.

With respect to the former, it could be interesting to use a shorter value for T_{tup} during periods of intense team dynamism, while a larger value could be adequate for periods of quietness, profiting from less transmissions. The on-line adaptation of T_{tup} also requires a consensus procedure, similarly to the membership vector referred in Chapter 5. However, if there is a need for a sudden change from a large to a short T_{tup}, the nodes with such need can start transmitting asynchronously until the consensus is achieved.

Concerning ϵ, note that a smaller value will make the protocol adapt less, thus keeping the actual round period closer to T_{tup} while a larger value allows stronger T_{tup} fluctuations. Thus, in an attempt to improve the regularity of the round time without giving away the adaptivity, one could think of a dynamic value for ϵ keeping it low for periods of sporadic larger delays, which would be filtered out, and enlarging it upon persistence of larger network delays.

Stochastic analysis of the time to join

In Chapter 5 we have provided guidelines to configure our protocol according to the expected bandwidth of external interfering traffic and the communication requirements of the team. However, we left the computation of the $n(b, p)$ function in Equations 5.14 and 5.15 for future work. This function gives us the number of extra rounds that need to be considered in the joining of a new node during a reconfiguration given a bit error rate b and a consequent probability p of successful packet reconfiguration. This requires stochastic analysis that still needs to be done. A measurement of the distribution of this function in a specific operational scenario was already shown in Chapter 7.

Generalization and dynamism of the RTDB

During the design of the RTDB several options had to be taken and two of them were considered from the beginning as aspects for future improvement. These are related to the dependence on the computing platform and the static configuration.

Currently, the RTDB uses a memory copy model of data transfer that requires homogeneous hardware architectures and similar data types. Thus, one desirable feature would be to use abstract data types so that heterogeneous computing platforms could be integrated. This seems relatively simple to implement.

On the other hand, it would also be desirable to circumvent the static configuration and allow the dynamic creation of RTDB items. This is a more challenging feature that would allow, for example, dynamically integrating robots that were developed separately but providing services that could be useful to others in specific contexts. How to announce both new services and service needs, and how to compose dynamically collaborative behaviors out of such independently developed services seems a rather interesting line of research.

Bibliography

[1] W.H.T.M. Aangenent, J.J.T.H. de Best, B.H.M. Bukkems, F.M.W. Kanters, K.J. Meessen, J.J.P.A Willems, R.J.E. Merry, and M.J.G. v.d. Molengraft. Tech United Eindhoven - Team Description Paper. Technical report, Technical University of Eindhoven, 2009.

[2] Norman Abramson. THE ALOHA SYSTEM: Another Alternative for Computer Communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), pages 281–285, New York, USA, 1970. ACM.

[3] AirMagnet. 802.11n Primer. Technical report, AirMagnet Inc., August 2008.

[4] H. Levent Akin, Andreas Birk, Andrea Bonarini, Gerhard Kraetzschmar, Pedro Lima, Daniele Nardi, Enrico Pagello, Monica Reggiani, Alessandro Saffiotti, Alberto Sanfeliu, and Matthijs Spaan. Two "Hot Issues" in Cooperative Robotics: Network Robot Systems, and Formal Models and Methods for Cooperation. Technical report, EURON Special Interest Group on Cooperative Robotics, 2008.

[5] Luís Almeida and Paulo Pedreiras. Scheduling Within Temporal Partitions: Response-time Analysis and Server Design. In *Proceedings of the 4th ACM International Conference on Embedded Software*, EMSOFT '04, pages 95–103, Pisa, Italy, 2004. ACM.

[6] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku, and Woo-Keun Yoon. RT-Middleware: Distributed Component Middleware for RT (Robot Technology). In *Proceedings of the IROS 2005 - IEEE/RSJ International Conference on Intelligent Robots and System*, pages 3933–3938, Alberta, Canada, August 2005.

[7] José L. Azevedo, Cristóvão Cruz, João Cunha, Manuel B. Cunha, Nuno Lau, Ciro Martins, António J. R. Neves, Eurico Pedrosa, Artur Pereira, António J. S. Teixeira, and Mário Antunes. CAMBADA@Home 2011 – Team Description Paper. Technical report, University of Aveiro, 2011.

[8] José Luís Azevedo, Bernardo Cunha, and Luís Almeida. Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. In *Proceedings of the ETFA 2007 - IEEE Conference on Emerging Technologies and Factory Automation*, pages 973–980, September 2007.

[9] Ali Balador and Ali Movaghar. The Novel Contention Window Control Scheme for IEEE802.11 MAC Protocol. In *Proceedings of the NSWCTC 2010 - 2nd International Conference on Networks Security, Wireless Communications and Thrusted Computing*, Wuhan, China, April 2010.

[10] Paulo Bartolomeu, José Alberto Fonseca, and Francisco Vasques. Implementing the wireless ftt protocol: A feasibility analysis. In *Proceedings of the ETFA 2010 - IEEE Conference on Emerging Technologies and Factory Automation*, pages 1–10, Bilbao, Spain, September 2010.

[11] Moris Behnam, Thomas Nolte, and Nathan Fisher. On optimal real-time subsystem-interface generation in the presence of shared resources. In *Proceedings of the ETFA2010 - IEEE Conference on Emerging Technologies and Factory Automation*, pages 1–8, Bilbao, Spain, September 2010.

[12] Bo Bernhardsson, Johan Eker, and Joakim Persson. Bluetooth in Control. In Dimitrios Hristu-Varsakelis and William Levine, editors, *Handbook of Networked and Embedded Control Systems*, Control Engineering, pages 699–720. Birkhäuser Boston, 2005.

[13] Chiara Buratti, Andrea Conti, Davide Dardari, and Roberto Verdone. An Overview on Wireless Sensor Networks Technology and Evolution. *Sensors*, 9(9):6869–6896, August 2009.

[14] Giorgio C. Buttazzo, Marko Bertogna, and Gang Yao. Limited Preemptive Scheduling for Real-Time Systems. A Survey. *IEEE Transactions on Industrial Informatics*, 9(1):3–15, February 2013.

[15] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. In Ronald C. Arkin and George A. Bekey, editors, *Robots Colonies*, volume 4 of *Autonomous Robots*, pages 7–27. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[16] Eric Colon. Annex G - Survey of Software Frameworks for Robotics Applications. Technical Report RTO-TR-IST-032, Robotics Laboratory – Royal Military Academy, January 2002.

[17] Daniel D Corkill. Collaborating Software: Blackboard and Multi-Agent Systems & the Future. In *Proceedings of the International Lisp Conference*, New York, USA, October 2003.

[18] Robson Costa, Paulo Portugal, Francisco Vasques, and Ricardo Moraes. A TDMA-based mechanism for real-time communication in IEEE 802.11e networks. In *Proceedings of the ETFA 2010 - IEEE Conference on Emerging Technologies and Factory Automation*, pages 1–9, Bilbao, Spain, September 2010.

[19] Crossbow. *MICAz: Wireless Measurement System.*

[20] Edward Curry. Message-Oriented Middleware. In Qusay H. Mahmoud, editor, *Middleware for Communications*, chapter 1, pages 1–28. John Wiley and Sons, Chichester, England, 2004.

[21] Cisco Validating Design. Enterprise Mobility 7.3 Design Guide. Technical report, Cisco Systems, Inc., September 2013.

[22] Nuno Alexandre Neto Dias. Supervisão, Comando e Controlo para Sistemas Robóticos. Master's thesis, Departamento de Engenharia Electrotécnica, Instituto Superior de Engenharia do Porto, Porto, October 2009.

[23] Stefan Enderle, Hans Utz, Stefan Sablatnog, Steffen Simon, Gerhard Kraetzschmar, and Gunther Palm. Miro: Middleware for Autonomous Mobile Robots. In *Proceedings of the IFAC-01 - Conference on Telematics Applications in Autonomous and Robotics*, Weingarten, Germany, July 2001.

[24] Mustafa Ergen. IEEE 802.11 Tutorial. Technical report, Department of Electrical Engineering and Computer Science, University of California Berkeley, June 2002.

[25] Extricom. 802.11n for Enterprise Wireless LANs. Technical report, Extricom Ltd, 2010.

[26] Hugo Ferreira, Alfredo Martins, André Dias, Carlos Almeida, José M. Almeida, and Eduardo P. Silva. ROAZ Autonomous Surface Vehicle Design and Implementation. *Robótica - Controlo, Automação e instrumentação*, 2 trimestre(67), July 2007.

[27] Erian Ferro and Francesco Potortí. Bluetooth and Wi-Fi Wireless Protocols: a Survey and a Comparison. *IEEE Wireless Communications*, 12(1):12–26, February 2005.

[28] Nuno M. Figueiredo, António J. R. Neves, Nuno Lau, Artur Pereira, and Gustavo Corrente. Control and Monitoring of a Robotic Soccer Team: The Base Station Application. In Luís Seabra Lopes, Nuno Lau, Pedro Mariano, and Luís M. Rocha, editors, *Progress in Artificial Intelligence*, volume 5816 of *Lecture Notes in Computer Science*, pages 299–309. Springer Berlin Heidelberg, 2009.

[29] A. Flammini, D. Marioli, E. Sisinni, and A. Taroni. A real-time Wireless Sensor Network for temperature monitoring. In *Proccedings of the ISIE 2007 - IEEE International Symposium on Industrial Electronics*, pages 1916–1920, June 2007.

[30] David Flanagan, Jim Farley, William Crawford, and Kris Magnusson. *Java Enterprise In a Nutshell*. O'Reilly and Associates, Inc., 1999.

[31] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc., April 2002.

[32] Aniruddha Gokhale and Douglas C. Schmidt. Techniques for optimizing CORBA middleware for distributed embedded systems. In *Proceedings of the IEEE INFOCOM '99 - Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 513–521, New York, USA, March 1999.

[33] Nada Golmie and Nicolas Chevrollier. Techniques to improve Bluetooth performance in interference environments. In *Proceedings of the MILCOM 2001 - Military Communications Conference. Communications for Network-Centric Operations: Creating the Information Force*, volume 1, pages 581–585, October 2001.

[34] IEEE 802.11 Working Group. IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirement. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical report, IEEE, Inc, February 2012.

[35] IEEE 802.15 Working Group. IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirement. Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs). Technical report, IEEE, Inc, June 2005.

[36] IEEE 802.15 Working Group. IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirement. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, IEEE, Inc, June 2011.

[37] Object Management Group. Data Distribution Service for Real-time Systems, Version 1.2. Technical Report formal/07-01-01, Object Management Group, Inc., 2007.

[38] Object Managment Group. The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification (DDS-RTPS), Version 2.1. Technical Report formal/2010-11-01, Object Management Group, Inc., 2010.

[39] Jaap Haartsen and Sven Mattisson. Bluetooth - A New Low-Power Radio Interface Providing Short-Range Connectivity. *Proceedings of the IEEE*, 88(10):1651–1661, October 2000.

[40] Ehsan Haghani, Michael N. Krishnan, and Avideh Zakhor. Adaptive Carrier-Sensing for Throughput Improvement in IEEE 802.11 Networks. In *Proceedings of the GLOBECOM 2010 - IEEE Global Communications Conference Exhibition and Industrial Forum*, Miami, Florida, December 2010.

[41] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, July 1985.

[42] Michi Henning. A New Approach to Object-Oriented Middleware. *IEEE Internet Computing*, pages 66–75, January/February 2004.

[43] Dimitrios Hristu-Varsakelis and William S. Levine, editors. *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[44] Tran Duy Khanh, Zidek Martin, Benda Jan, Kubias Jiri, and Sojka Michal. Autonomous robot running Linux for the Eurobot 2007 competition. In *Proceeding of the 9th Real-Time Linux Workshop*, Linz, Austria, November 2007.

[45] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The Robot World Cup Initiative, 1995.

[46] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The Robot World Cup Initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, USA, 1997. ACM.

[47] Kenneth J. Klingenstein. Middleware: The Second Layer of IT Infrastructure. *CAUSE/EFFECT journal*, 22(4), 1999.

[48] Hermann Kopetz and Günther Bauer. The Time-Triggered Architecture. *Proceedings of the IEEE*, 91(1):112–126, January 2003.

[49] Hermann Kopetz and Johannes Reisinger. The non-blocking write protocol NBW: A solution to a real-time synchronization problem. In *Proceedings of the Real-Time Systems Symposium*, pages 131–137, December 1993.

[50] Nuno Lau, Luís S. Lopes, Gustavo Corrente, and Nuno Filipe. Multi-robot team coordination through roles, positionings and coordinated procedures. In *Proceedings of the IROS 2009 - IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5841–5848, St. Louis, MO, USA, October 2009.

[51] Nuno Lau, Luís Seabra Lopes, Gustavo Corrente, Nelson Filipe, and Ricardo Sequeira. Robot Team Coordination using Dynamic Role and Positioning Assignment and Role Based Setplays. *Mechatronics*, 21(2):445–454, 2011. Special Issue on Advances in intelligent robot design for the Robocup Middle Size League.

[52] Nuno Lau, Luís Seabra Lopes, and Gustavo A. Corrente. CAMBADA: Information Sharing and Team Coordination. In *Proceedings of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open – ROBOTICA*, pages 27–32, Aveiro, Portugal, April 2008.

[53] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In *Proceedings of the IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 46–51, November 2007.

[54] P.U. Lima. Robotics Educational Activities in Portugal: A Motivating Experience. *IEEE Robotics Automation Magazine*, 14(2):16–17, June 2007.

[55] Yabo Liu, Jianhua Yang, Yao Zheng, Zhaohui Wu, and Min Yao. Multi-Robot Coordination in Complex Environment with Task and Communication Constraints. *International Journal of Advanced Robotic Systems*, 10, 2013.

[56] Jianhua Ma. Computer Networks (Digital Communication and Networks) Course, Lecture Notes, Lecture 4. School of Computer and Information Sciences, Hosei University, Tokyo, Japan.

[57] Ricardo Jorge Silva Machado. Ambiente de simulação 3D para um veículo de condução autónoma. Master's thesis, Universidade de Aveiro, Aveiro, Portugal, 2011.

[58] Stefan Mahlknecht and Marco Spinola Durante. WUR-MAC: Energy efficient wakeup receiver based MAC protocol. In *Proccedings of the IFAC-FeT 2009 - 8th International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, volume 8, Republic of Korea, May 2009.

[59] Alexei Makarenko, Alex Brooks, and Tobias Kaupp. Orca: Components for Robotics. In *Proceedings of the IROS 2006 - IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006.

[60] Alexei Makarenko, Alex Brooks, and Ben Upcroft. An Autonomous Vehicle using ICE and ORCA. Connections - ZeroC's Newsletter for the Ice Community, April 2007.

[61] Patrick Ferreira Marques. Concurrent architecture for control of an autonomous driving vehicle. Master's thesis, Universidade de Aveiro, Aveiro, Portugal, 2010.

[62] Tiago Meireles, José Fonseca, and Joaquim Ferreira. Vehicular Flexible Time-Triggered Protocol (V-FTT). Technical Report 01/2003, Embedded System Group - Telecommunications Institute, Aveiro, Portugal, March 2013.

[63] João Messias, Aamir Ahmad, João Reis, Miguel Serafim, and Pedro Lima. Socrob 2013 - team description paper. Technical report, Institute for Systems and Robotics, Instituto Superior Técnico, 2013.

[64] Nader Mohamed and Jameela Al-Jaroodi. Characteristics of Middleware for Networked Collaborative Robots. In *Proceedings of the CTS'2008 - International Symposium on Collaborative Technologies and Systems*, Irvine, CA, USA, May 2008.

[65] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Middleware for Robotics: A Survey. In *Proceedings of the RAM'08 - IEEE Conference on Robotics, Automation and Mechatronics*, pages 736–742, Chengdu, China, September 2008.

[66] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. A Review fo Middleware for Network Robots. *Proceedings of the IJCSNS - International Journal of Computer Science and Network Security*, 9(5):139–148, May 2009.

[67] Fred Mora. Bring an Atomic Clock to Your Home with Chrony. *Linux Journal*, 2002(101), September 2002.

[68] António J. R. Neves, José Luís Azevedo, Bernardo Cunha, Nuno Lau, João Silva, Frederico Santos, Gustavo Corrente, Daniel A. Martins, Nuno Figueiredo, Artur Pereira, Luís Almeida, Luís Seabra Lopes, Armando J. Pinho, João Rodrigues, and Paulo Pedreiras. CAMBADA Soccer Team: from Robot Architecture to Multiagent Coordination. In Vladan Papi, editor, *Robot Soccer*, pages 19–45. InTech, 2010.

[69] António J. R. Neves, Gustavo A. Corrente, and Armando J. Pinho. An Omnidirectional Vision System for Soccer Robots. In António José Neves, Manuel Filipe Santos, and Jos Manuel Machado, editors, *Progress in Artificial Intelligence*, volume 4874 of *Lecture Notes in Computer Science*, pages 499–507. Springer Berlin Heidelberg, 2007.

[70] António J. R. Neves, Nuno Lau, Luís Paulo Reis, Bruno Pimentel, João Silva, Alina Trifan, Nima Shafii, Vasco Santos, Sanaz Zadegan, and Sanaz Bahmankhah. Portuguese Team: Team Description Paper for RoboCup 2013. Technical report, University of Aveiro, University of Porto and University of Minho, 2013.

[71] António J. R. Neves, Daniel A. Martins, and Armando J. Pinho. A hybrid vision system for soccer robots using radial search lines. In *Proceedings of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open – ROBOTICA*, pages 51–55, Aveiro, Portugal, April 2008.

[72] Luís Oliveira, Luís Almeida, and Frederico Santos. A Loose Synchronisation Protocol for Managing RF Ranging in Mobile Ad-Hoc Networks. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranlı, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 574–585. Springer Berlin Heidelberg, 2012.

[73] Luís Oliveira, Hongbin Li, and Luís Almeida. Experiments with navigation based on the rss of wireless communication. In *ROBOTICA2010: 10th Conference on Mobile Robots and Competitions*, 2010.

[74] Object Management Group (OMG). Real-time CORBA Specification, Version 1.2. Technical Report formal/05-01-04, Object Management Group, Inc., 2005.

[75] Object Management Group (OMG). Common Object Request Broker Architecture (CORBA) for *embedded* Specification), Version 1.0. Technical Report formal/2008-11-06, Object Management Group, Inc., 2008.

[76] Object Management Group (OMG). Robotic Technology Component (RTC), Version 1.1. Technical report, Object Management Group, Inc., 2012.

[77] ISO Technical Committee on Information Technology (ISO/IEC JTC1) and the Object Management Group (OMG). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3. Technical Report formal/2012-11-12, Object Management Group, Inc., 2012.

[78] Bernardo Ordoñez. *Estratégia de controle cooperativo baseado em consenso para um grupo multi-veículos*. PhD dissertation, Universidade Federal de Santa Catarina, Florianópolis, Brazil, May 2013.

[79] Gerardo Pardo-Castellote. OMG Data-Distribution Service: Architectural Overview. In *Proceedings of the ICDCSW'03 - 23rd International Conference on Distributed Computing Systems Workshops*, pages 200–206, Providence, Rhode Island, USA, May 2003.

[80] Gerardo Pardo-Castellote. OMG Data Distribution Service: Real-Time Publish/Subscribe Becomes a Standard. *RTC Magazine*, January 2005.

[81] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheller, and Andrew Ng. ROS: an open-source Robot Operating System. In *Proceedings of the ICRA'09 - IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[82] Krithi Ramamritham, Sang H. Son, and Lisa Cingiser DiPippo. Real-Time Databases and Data Services. *Real-Time Systems*, 28(2-3):179–215, 2004.

[83] Inc. Real-Time Innovations. *RTI Connext DDS Professional*, 2013.

[84] Luís Paulo Reis, Nuno Lau, and Eugénio Oliveira. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In Markus Hannebauer, Jan Wendler, and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, volume 2103 of *Lecture Notes in Computer Science*, pages 175–197. Springer, 2001.

[85] Frederico Santos and Luís Almeida. On the effectiveness of IEEE802.11 broadcasts for soft real-time communication. In *Proceedings of the RTN'05 - 4th International Workshop on Real-Time Networks*, Palma de Mallorca, Spain, July 2005.

[86] Frederico Santos, Luís Almeida, Paulo Pedreiras, Luís Seabra Lopes, and Tullio Facchinetti. An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents. In *Proceedings of the WACERTS'04 - Workshop on Architectures for Cooperative Embedded Real-Time Systems, Satellite event of the RTSS 2004 - 25th IEEE International Real-Time Systems Symposium*, Lisbon, Portugal, December 2004.

[87] Frederico Santos, Gustavo Currente, Luís Almeida, Nuno Lau, and Luís S. Lopes. Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. In *Proceedings of the 2nd Workshop on Intelligent Robotics (IRobot 2007), Satellite event of the EPIA 2007 - 13th Portuguese Conference on Artificial Intelligence*, Guimarães, Portugal, December 2007.

[88] Douglas Schmidt, David Levine, and Sumedh Mungee. The design of the TAO real-time object request broker. *Computer Communications*, 21(4):294–324, 1998.

[89] Domenico Sicignano. *Analysis, evaluation and improvement of RT-WMP for real-time and QoS wireless communication: Applications in confined environments.* PhD dissertation, University of Zaragoza, Zaragoza, Spain, March 2013.

[90] Bluetooth SIG. Specification of the Bluetooth System version 4.1. Technical report, Bluetooth SIG, Inc., December 2013.

[91] Axel Sikora and Voicu F. Groza. Coexistence of IEEE802.15.4 with other Systems in the 2.4 GHz-ISM-Band. In *Proceedings of the IMTC 2005 - IEEE Instrumentation and Measurement Technology Conference*, volume 3, pages 1786–1791, May 2005.

[92] Valter Silva, Ricardo Marau, Luís Almeida, Joaquim Ferreira, Mário Calha, Paulo Pedreiras, and José Fonseca. Implementing a distributed sensing and actuation system: The CAMBADA robots case study. In *Proceedings of the ETFA 2005 - 10th IEEE Conference on Emerging Technologies and Factory Automation*, volume 2, pages 781–788, Catania, Italy, September 2005.

[93] Hendrick Skubch. *Modelling and Controlling of Behaviour for Autonomous Mobile Robots.* Springer Vieweg, 2012.

[94] Petr Smolik and Pavel Pisa. ORTE: The Open Real Time Ethernet. In *Proceedings of the RTN'08 - 7th International Workshop on Real-Time Networks*, Prague, Czech Republic, July 2008.

[95] IEEE Vehicular Technology Society. IEEE Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages. Technical report, IEEE, Inc, April 2013.

[96] Peter Stone and Manuela Veloso. Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[97] Danilo Tardioli and José Luis Villarroel. Real Time Communications over 802.11: RT-WMP. In *Proceedings of the MASS 2007 - 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 1–11, Pisa, Italy, October 2007.

[98] Yu Tian, Yang Lv, and Ling Tong. Design and application of sink node for wireless sensor network. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, 32(2):531–544, 2013.

[99] Hans Utz, Stefan Sablatnog, Stefan Enderle, and Gerhard Kraetzschmar. Miro: Middleware for Mobile Robot Applications. *IEEE Transactions on Robotics and Automation*, 18(4):493–497, August 2002.

[100] Thilo Weigel, Jens-Steffen Gutmann, Markus Dietl, Alexander Kleiner, and Bernhard Nebel. CS Freiburg: coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, 18(5):685–699, Oct 2002.

[101] Andreas Willig, Kirsten Matheus, and Adam Wolisz. Wireless Technology in Industrial Networks. *Proceedings of the IEEE*, 93(6):1130–1151, June 2005.

[102] Victor Fay Wolfe, Lisa Cingiser DiPippo, Roman Ginis, Michael Squadrito, Steven Wohlever, Igor Zykh, and Russell Johnston. Real-time CORBA. In *Proceedings of the 3rd IEEE Real Time Technology and Applications Symposium*, pages 148–157, Montreal, Quebec, Canada, June 1997.

[103] ZeroC, Inc. *ICE Manual - Distributed Programming with ICE, Version 3.5.1*, 2013.

[104] Hubbert Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.

Annex A

Network Occupancy

The medium occupancy in a IEEE 802.11 network is non-trivial to compute and the exact value is even impossible to obtain since the medium is shared, uncontrolled and the link may suffer disruption, interference of different sources, and even the transmission speed may vary dynamically.

Nevertheless, in this Annex we present an estimation that is valid under certain operational assumptions. We start by introducing some basic background that is needed to carry out the computations and then we present specific formulae that provide the desired estimation. In particular, we explain the inner computations of two functions used in Chapter 5, namely $nodeLoadTime()$ and $extLoadOcup()$.

A.1 IEEE 802.11 communications in infrastrutured mode

In IEEE 802.11 operating in infrastrutured mode, i.e., infrastructured, communication between nodes can be divided in three types (Figure A.1):

- **Unicast** is a one-to-one transmission method in which the network carries a packet to one receiver. In the unicast method, when multiple receivers require the same information from a sender, a similar data packet has to be sent multiple times. The guarantee of successful delivery is provided by an acknowledgment that is sent from the receiver back to the sender. If the acknowledgement is not received in a defined period of time, the sender retransmits the packet.

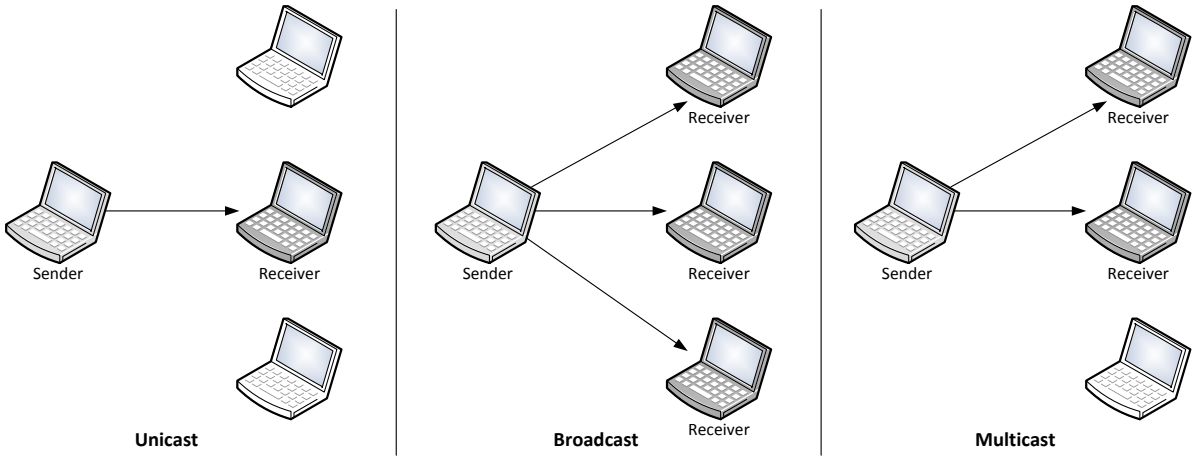


Figure A.1: Transmission methods

- **Broadcast** is a one-to-all transmission method in which the network carries a packet to all devices at the same time. Broadcast packets are sent to every node on the network and are not filtered or blocked by a router. Due to the nature of this method, it is impossible to provide a simple acknowledgment service;
- **Multicast** is a one-to-many transmission method in which the network carries a packet to multiple receivers at the same time. Multicast is similar to broadcasting, except that multicasting means sending to a specific group, whereas broadcasting implies sending to everybody, whether they want the traffic or not. This method does not provide acknowledgment, either.

In IEEE 802.11 networks working in infrastrutured mode, all the communications go through the AP, as explained in Section 2.1. A transmission from sender node A to destination node B involves in fact two transmissions. Initially, node A transmits to the AP and finally the AP retransmits to the destination node B. In a unicast interaction, each of these two transmissions is acknowledged (Figure A.2). A broadcast or multicast interaction with origin in node A is transformed in a unicast transmission from node A to the AP, i.e., acknowledged, and then the AP retransmits in broadcast/multicast fashion, respectively (Figure A.3). Note that when using a shared medium like the air, all types of interaction use the medium in an exclusive way, i.e., parallel transmissions are not possible.

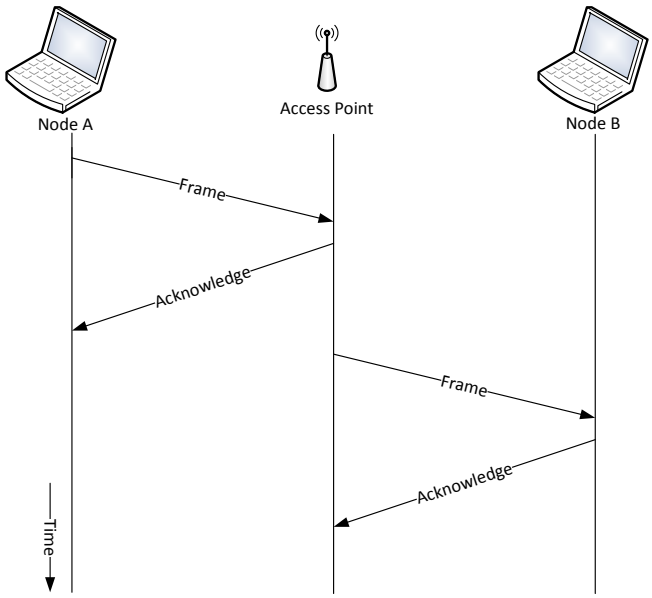


Figure A.2: Unicasts in IEEE 802.11 infrastrutured mode

A.2 IEEE 802.11 timings

As presented in Section 2.1, there are distinct IEEE 802.11 network types. The use of two frequency bands, $2.4Ghz$ and $5GHz$, creates the IEEE 802.11b/g and IEEE 80211a types, respectively.

In a simplified way, considering only the operation under the Distributed Coordination Function (DCF), before transmitting each node must sense the medium during a specified amount of time, namely Short Inter Frame Spacing (SIFS) or Distributed Coordination Function Inter Frame Spacing (DIFS). The former is used for higher priority transmissions, like acknowledgment packets. The latter applies to data packet transmissions (Figure A.4).

If the channel is sensed busy, then the node must wait until the channel is sensed free again and then wait for DIFS plus a random backoff interval. The backoff interval T_{BO} is used to reduce collisions among multiple nodes trying to transmit at the same time and is determined as:

$$T_{BO} = random(0, CW) \times t_{slot} \tag{A.1}$$

$$CW = 2^n - 1 \tag{A.2}$$

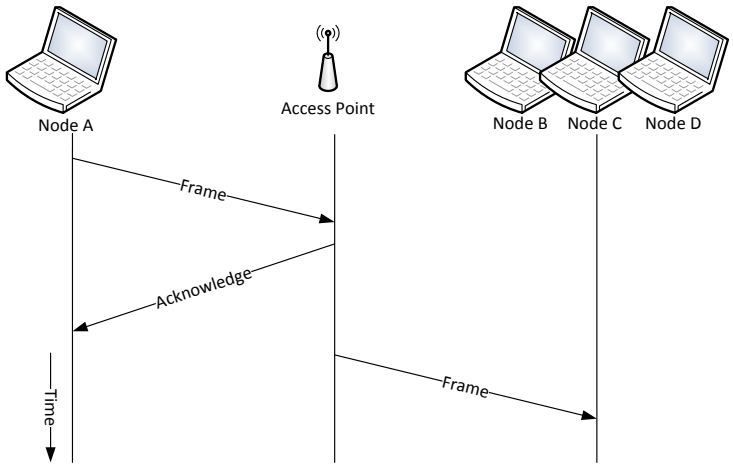


Figure A.3: Broadcasts and multicasts in IEEE 802.11 infrastrutured mode

The Contention Window (CW) parameter is adapted dynamically depending on collision occurrences. Each time a collision occurs, n is incremented. The minimum and maximum values of n are shown in Table A.1.

		802.11b	802.11g mixed	802.11g only g	802.11a
t_{slot}	(μs)	20	20	9	9
T_{SIFS}	(μs)	10	10	10	16
T_{DIFS}	(μs)	50	50	28	34
CW	n_{max}	10	10	10	10
	n_{min}	5	5	4	4
$T_{preamble}$	(μs)	192	192	26	20
$maxBitRate$	$(Mbps)$	11	54	54	54
$avrBitRate$	$(Mbps)$	5.5	24	24	24
$ackBitRate$	$(Mbps)$	2	2	24	24
$multiBitRate$	$(Mbps)$	1	1	6	6

Table A.1: IEEE802.11 network parameters

Each IEEE 802.11 network type defines a group of transmission bit rates as previously presented in Table 2.1. The bit rate selection is automatically adjusted by the transmitting node and is a function of the link quality. However, there are other causes that affect the bit rate. For instance, each time a transmission in not correctly acknowledged the retransmission is carried out at the next lower bit rate.

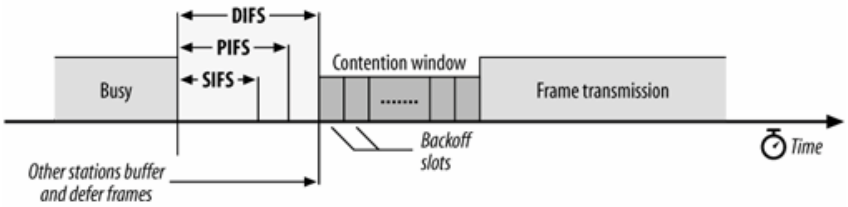


Figure A.4: Interframe spacing relationships [31]

The transmission mode also limits the bit rate. Unicast transmissions can be carried out at any available bit rate, being $maxBitRate$ the maximum allowed. Since it is practically impossible to predict the actual bit rate that will be used, we opted for using an average value ($avrBitRate$) in the following computations. On the other hand, acknowledgements have a specific bit rate ($ackBitRate$) and the broadcast or multicast frames use a lower value ($multiBitRate$) to increase their robustness and consequently the chances that they arrive at all nodes in the AP neighborhood, independently of the link quality.

Table A.1 presents a summary of these parameters for each network type. Note that g type is retro-compatible with b type. In fact, in a g network, each time a b node registers, all the other nodes must revert to the b type timing values to ensure compatibility, even if they could otherwise continue transmitting data using g higher bit rates.

The computation of the payload transmission time depends on the number of payload bytes and the bit rate in use. For IEEE 802.11b, using High Speed Direct-Sequence Spread Spectrum (HS/DSSS) modulation, the relation between both values gives the amount of time required. But, IEEE 802.11a/g use a distinct modulation technique (OFDM) with large amount of bits per transmission symbol, as presented in Table A.2. Each symbol, independently of the bit rate in use, has a fixed duration of $4\mu s$.

Bit rate ($Mbps$)	# bits/symbol
6	24
9	36
12	48
18	72
24	96
36	144
48	192
54	216

Table A.2: Encoding for OFDM data rates [31]

A.3 Function *nodeLoadTime()*

$nodeLoadTime(D_i, frameType, netType)$ returns an approximate value of the amount of time that the network will be occupied by a transmitter node, being:

- D_i – The maximum number of data bytes that the node could transmit in a round;
- $frameType$ – The communication type in use, that can be:
  - u – Unicast;
  - m – Multicast;
  - b – Broadcast.
- $netType$ – The IEEE 802.11 type in use, which can be:
  - a – IEEE 802.11a;
  - b – IEEE 802.11b;
  - g – IEEE 802.11g.

As shown in Figures A.2 and A.3 the type of communication used defines the number of exchanged frames, leading to (u means unicast type):

$$nodeLoadTime(D_i, frameType, netType) = \begin{cases} T_{unicast}(D_i, netType) & : frameType = u \\ T_{multicast}(D_i, netType) & : frameType \neq u \end{cases} \tag{A.3}$$

A unicast transmission from node A to node B is in fact composed by two data frames and respective acknowledgements:

$$\begin{aligned} T_{unicast}(D_i, netType) = {} & T_{frame}(D_i, avrBitRate(netType), netType) + T_{ack}(netType) + \\ & + T_{frame}(D_i, avrBitRate(netType), netType) + T_{ack}(netType) \end{aligned} \tag{A.4}$$

For multicast or broadcast transmissions, the transmission time is estimated as:

$$\begin{aligned} T_{multicast}(D_i, netType) &= T_{broadcast}(D_i, netType) \\ &= T_{frame}(D_i, avrBitRate(netType), netType) + T_{ack}(netType) + \\ &\quad + T_{frame}(D_i, multiBitRate(netType), netType) \end{aligned} \tag{A.5}$$

Each data frame transmission then takes the duration of DIFS (T_{DIFS}), a backoff time (T_{BO}), a fixed duration of the preamble plus other additional data (sum and Physical Layer Convergence Protocol (PLCP)) ($T_{preamble}$) and the payload data ($T_{payload}$) transmited at a designated *bitRate*:

$$\begin{aligned} T_{frame}(D_i, bitRate, netType) = T_{DIFS}(netType) + T_{BO}(netType) + T_{preamble}(netType) + \\ + T_{payload}(D_i, bitRate, netType) \end{aligned} \tag{A.6}$$

On the other hand, acknowledge frames are given high priority through the use of SIFS, the preamble time ($T_{preamble}$) and the payload time corresponding to 12 bytes at *ackBitRate* bit rate:

$$\begin{aligned} T_{ack}(netType) = T_{SIFS}(netType) + T_{preamble}(netType) + \\ + T_{payload}(12, ackBitRate(netType), netType) \end{aligned} \tag{A.7}$$

To complete equations A.6 and A.7 we need to compute the time required to transmit D bytes of payload at bit rate *bitRate*, using the *netType* IEEE 802.11 type:

$$T_{payload}(D, bitRate, netType) = \begin{cases} \dfrac{D \times 8}{bitRate} & : netType = b \\ \left\lceil \dfrac{D \times 8}{bitsPerSymbol(bitRate)} \right\rceil \times 4\mu s & : netType \neq b \end{cases} \tag{A.8}$$

The *bitsPerSymbol*() function returns the number of bits per symbol directly from Table A.2.

A.4 Function *extLoadOcup()*

$extLoadOcup(L, frameType, netType)$ returns an approximate value of the network occupancy corresponding to a certain amount of uncontrolled external traffic, given by the required throughput L in *Mbps*, using frames of $frameType$ and a $netType$ IEEE 802.11 type of network, with the options described in Section A.3.

Avoiding the knowledge of which payload size is used in the frames corresponding to L, we adopt the same approach of using an average value. Given the typical MTU of $1500B$, the corresponding average payload size D is $750B$. Following the same reasoning as in the previous section, then we can directly expressed the desired function as:

$$
\begin{aligned}
&extLoadOcup(L, frameType, netType) = \\
&\qquad = \begin{cases} \left\lceil \dfrac{L}{750 \times 8} \right\rceil \times T_{unicast}(750, netType) & : frameType = u \\ \\ \left\lceil \dfrac{L}{750 \times 8} \right\rceil \times T_{multicast}(750, netType) & : frameType \neq u \end{cases}
\end{aligned}
\tag{A.9}
$$

Annex B

Configuration parameters used by CAMBADA

The CAMBADA team of robotic soccer robots was not only a motivation but also the scope within which most of this work was carried out. Moreover, this team was used in several of the validation experiments referred along this document, and it is also one of the main users of the RTDB middleware with the Reconfigurable and Adaptive TDMA wireless protocol.

Therefore, it is illustrative and also relevant to address the configuration used in both communications protocol and middleware. The former configuration parameters are defined in Table B.1.

Update period	T_{tup}	$100ms$
Communication time	T_{wt} @ IEEE802.11a T_{wt} @ IEEE802.11b/g	$1ms$ $4ms$
Ratio transmission window used for sync	ϵ	$\frac{2}{3}$
Max. number of agents	$max\{N\}$	10
Max. number of items in RTDB	$max\{ITEM\}$	100

Table B.1: Configuration parameters

The RTDB configuration is shown in Listing B.1, encompassing one base station and six robots. Note the similar configuration of all the robots, which is justified by the dynamic role assignment policy used by the team, meaning that any robot can play in any position in the field. Moreover, both the software and hardware architectures are identical in all robots, thus leading to similar RTDB items.

```
AGENTS = BASE_STATION, CAMBADA_1, CAMBADA_2, CAMBADA_3, CAMBADA_4, CAMBADA_5,
         CAMBADA_6;

ITEM ROBOT_WS { datatype = RobotWS; headerfile = RobotWS.h; }
ITEM LAPTOP_INFO { datatype = LaptopInfo; headerfile = SystemInfo.h; }
ITEM COACH_INFO { datatype = CoachInfo; headerfile = CoachInfo.h; }
ITEM VISION_INFO { datatype = VisionInfo; headerfile = VisionInfo.h; }
ITEM FRONT_VISION_INFO { datatype = FrontVisionInfo; headerfile = VisionInfo.h; }
ITEM FORMATION_INFO { datatype = FormationInfo; headerfile = CoachInfo.h; }

ITEM CMD_VEL { datatype = CMD_Vel; headerfile = HWcomm_rtdb.h; }
ITEM CMD_POS { datatype = CMD_Pos; headerfile = HWcomm_rtdb.h; }
ITEM CMD_KICKER { datatype = CMD_Kicker; headerfile = HWcomm_rtdb.h; }
ITEM CMD_INFO { datatype = CMD_Info; headerfile = HWcomm_rtdb.h; }
ITEM CMD_HWERRORS { datatype = CMD_HWerrors; headerfile = HWcomm_rtdb.h; }
ITEM CMD_GRABBER { datatype = CMD_Grabber; headerfile = HWcomm_rtdb.h; }
ITEM LAST_CMD_VEL { datatype = CMD_Vel; headerfile = HWcomm_rtdb.h; }
ITEM REMOTE_CMD { datatype = RemoteCMD; headerfile = rtdb_remoteControl.h; }
ITEM CMD_IMU { datatype = CMD_Imu; headerfile = HWcomm_rtdb.h; }
ITEM CMD_SYNCIMU { datatype = int; headerfile = stdio.h; }
ITEM CMD_GRABBER_INFO { datatype = CMD_Grabber_Info; headerfile = HWcomm_rtdb.h; }
ITEM CMD_GRABBER_CONFIG { datatype = CMD_Grabber_Config; headerfile = HWcomm_rtdb.h; }

SCHEMA BaseStation
{
    shared = COACH_INFO, FORMATION_INFO, REMOTE_CMD;
}

SCHEMA Player
{
    shared = ROBOT_WS, LAPTOP_INFO;
    local = COACH_INFO, VISION_INFO, FRONT_VISION_INFO, CMD_VEL, CMD_POS, CMD_KICKER,
            CMD_INFO, CMD_HWERRORS, CMD_GRABBER, LAST_CMD_VEL, CMD_IMU, CMD_SYNCIMU,
            CMD_GRABBER_INFO, CMD_GRABBER_CONFIG;
}

ASSIGNMENT { schema = BaseStation; agents = BASE_STATION; }
ASSIGNMENT { schema = Player;  agents = CAMBADA_1, CAMBADA_2, CAMBADA_3, CAMBADA_4,
                                        CAMBADA_5, CAMBADA_6; }
```

Listing B.1: The CAMBADA RTDB configuration