

**Embedded multi-core systems for
mixed criticality applications
in dynamic and changeable real-time environments**

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	D11.4 – Final demonstrator implementation and evaluation	
Work package	WP11	Internet of Things
Task / Use Case	T11.1, T11.2, T11.3, T11.4, T11.5	Multimedia communication WebRTC Open deterministic networks Autonomic home networking Ultralowpower high data rate communication Synchronized low-latency deterministic networks
Subtasks involved		
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto: werner.weber@infineon.com	
Deliverable responsible	Quobis Yudani Riobó, yudani.riobo@quobis.com	
Version number	v1.0	
Date	07/04/2017	
Status	Final	
Dissemination level	Public (PU)	

Copyright: EMC² Project Consortium, 2017

Authors

Participant no.	Part. short name	Author name	Chapter(s)
15D	QUOBIS	Yudani Riobó Iago Soto Elías Pérez	General structure and document editing. Publishable Executive Summary 1. Introduction 2. Multimedia communication WebRTC
02D	TTT	Martijn Rooker	3. Open deterministic and mixed-criticality networking
02G	AIT	Axel Weißenfeld Christoph Schmittner	3. Open deterministic and mixed-criticality networking
04C	IMA	Jakub Neburka	4. Autonomic home networking
08A	HUA	George Bravos	4. Autonomic home networking
15R	Ambar	Roberto García Ignacio Ara Sara Díez del Valle	4. Autonomic home networking 5. Ultra low power high data rate communication
03B	BlueIce	Henri Cloetens Leon Cloetens Abelardo González	5. Ultra low power high data rate communication
15O	SevenS	Jose Luis Gutierrez Javier Díaz	6. Synchronized low-latency deterministic networks
15P	Schneider Electric	Benito Caracuel	6. Synchronized low-latency deterministic networks
15P	Schneider Electric	Francisco Ramos	Review

Publishable Executive Summary

WP11 Internet of Things focuses in different scenarios where the focus of this project helps to develop new functionalities. This work package includes five use cases within the IoT domain that allow showing implementations like audio and video communications over WebRTC, open deterministic networks, automatic home networking, ultralow power high database communication, and synchronized low-latency deterministic networks. Deliverable “D11.3 Detailed design and first prototype” provided the final design of the systems defined to deploy the use cases accordingly to defined requirements. This deliverable, “D11.4 Final demonstrator implementation and evaluation” collects the results of the work carried out by the partners involved in this WP to provide the systems defined to show the use cases accordingly to defined requirements.

Table of contents

1.	Introduction	7
1.1	Objective and scope of the document	7
1.2	Structure of the deliverable report	7
2.	Use case Multimedia communication WebRTC.....	8
2.1	Use case introduction.....	8
2.2	Description of final demonstrator.....	8
	Browser internals	8
	Chromium.....	8
	How to see different process.....	9
	Maximum number of process	9
	Same domain.....	9
	Pages opened from Javascript.....	9
	Forcing new pages to use their own process	9
	Application level. Javascript.....	9
	Workers	10
	Web Components	10
2.3	Results	11
2.4	Conclusions.....	11
3.	Use case Open deterministic networks.....	12
3.1	Use case introduction.....	12
3.2	Description of final demonstrator.....	13
3.3	Results	14
3.4	Conclusions.....	16
4.	Use case Autonomic home networking.....	18
4.1	Use case introduction.....	18
4.2	Description of final demonstrator.....	19
4.3	Results	20
4.4	Conclusions.....	20
5.	Use case Ultralowpower high datarate communication	21
5.1	Use case introduction.....	21
5.2	Description of final demonstrator.....	22
5.3	Results	25
5.4	Conclusions.....	29
6.	Use case Synchronized low-latency deterministic networks	30
6.1	Use case introduction.....	30
6.2	Description of final demonstrator.....	30
	Industrial Control System.....	31
6.3	Results	32

Synchronization Accuracy & Low-latency.....	33
Scalability.....	33
Industrial Timing Compatibility	34
Single Point of Failure Avoidance: Redundancy	34
Safety & Security	35
6.4 Conclusions.....	36
7. References	37
8. Appendix	38
8.1 Abbreviations	38
9. APPENDIX	39
9.1 Chromium architecture	39
9.2 How Chromium display a web page.....	40
9.3 Example subWorkers.....	41
9.4 Other types of workers.....	42
Shared workers	42
Service workers.....	42
9.5 ServiceWorkers lifecycle.....	44
9.6 Service Worker use case.....	44
9.7 Web Components with workers	45
9.8 Testing	46
Raw computing capacity	46
Objective.....	46
Procedure.....	46
Results - Desktop	46
Results - Mobile.....	48
File transfer.....	50
Objective.....	50
How To.....	50
Results	50
10. Appendix: Distributed Passenger Tracking in Deterministic Networks.....	52
10.1 Description of final demonstrator.....	53
10.2 Results	54
10.3 Conclusions.....	55
11. Appendix: Use case 5 demonstrator devices.....	56
11.1 WR-Switch.....	56
11.2 WR-ZEN.....	56
11.3 WR-LEN.....	57
11.4 Saitel DR platform.....	57
11.5 Saitel DP platform	58

List of figures

Figure 1: Demonstrator devices involved and Demonstrator architecture.....	8
Figure 2: Separate DOM between page and worker.	10
Figure 3: State of web components.....	11
Figure 4: Alerting staffs in case of an incident (e.g. amount of people exceeds threshold) by utilizing a deterministic network of cameras, servers and clients.....	12
Figure 5: An alert is thrown if a person enters the lawn (right image).....	14
Figure 6: Camera 1 (sender) detects 4 passengers. Camera 2 (receiver) by itself detects only two passengers due to occlusion. Camera 2 receives the information that camera 1 detected 4 persons, which helps to resolve the occlusion.	15
Figure 7: Schedule synthesis example – (a) Network topology, (b) TT flows, (c) Feasible scheduling solution.....	16
Figure 8: Routing synthesis example – (a) TT and AVB flows, (b) Routing solutions, (c) Optimized routes.....	16
Figure 10: The Autonomic home use case overall design	18
Figure 11: Demonstrator	19
Figure 12: Hardware architecture	22
Figure 13: BLUSP architecture.....	23
Figure 14: Software architecture.....	24
Figure 15: FPGA platform	25
Figure 16: Complete solution	26
Figure 17: Ultra low power high data rate communication receive sensitivity.....	27
Figure 18: Digital current profile.....	28
Figure 19: Commercial product current profiles.....	28
Figure 20: Design of a heterogeneous synchronized low-latency deterministic redundant network using WR-PTP, PTP and IRIG-B	31
Figure 21: Industrial Control System for final demonstrator	32
Figure 22: Final demonstrator of the heterogeneous synchronized low-latency deterministic redundant network using WR-PTP, PTP and IRIG-B	33
Figure 23: WR-LEN daisy-chain synchronization results	34
Figure 24: Synchronization sources. PTP-enabled devices (left), IRIG-B-enabled devices (right)	34
Figure 25: HSR protocol design (left) and implementation (right)	35
Figure 26: WP6 tool – RTU Fault Tree Analysis (FTA).....	35
Figure 27: Multi process architecture.....	39
Figure 28: Life cycle of a ServiceWorker.	44
Figure 29: People A and B are occluded in the view of camera 1, while not occluded in camera 2.	52
Figure 30: Setup of Multi-Camera Passenger Detection Using Open Deterministic Network.	53
Figure 31: Overview of the proposed passenger tracking.	53
Figure 32: Algorithm flow of people detection for individual camera.....	54
Figure 33: Camera A detects 4 persons in frame 376 and 3 persons in frame 732. This metadata helps camera B, which locally only detects two persons in both frames, to resolve the occlusion of persons.....	55

1. INTRODUCTION

1.1 Objective and scope of the document

This document reports the results of the implementation of the demonstrators for the five use cases in WP11. The objective is to provide a description of the final demonstrators deployed to test the use cases described in “D11.3 Detailed design and first prototype” and evaluate the results according to the objectives.

1.2 Structure of the deliverable report

This document is organized as follows: the sections 2, 3, 4, 5 and 6 present the five use cases in WP11. Each of these sections reports the implementation of the final demonstrators to evaluate the results of the project for the particular use cases. Section 7 lists the references. Finally, a collection of appendices provide additional details about the use cases.

2. USE CASE MULTIMEDIA COMMUNICATION WEBRTC

2.1 Use case introduction

This use case addresses large-scale application of Unified Communication Services using HTML5 based Web Browsers on Embedded Systems. A device of the family of Android TVs was selected to act as one of the web points with the possibility to add multimedia peripherals like camera and microphone. The application is a peer-to-peer videoconferencing solution between two devices which has been explained in “*D11.3 Detailed design and first prototype*”.

2.2 Description of final demonstrator

Based on the results collected in “*D11.2 Preliminary system design*” and “*D11.3 Detailed design and first prototype*”, the final design for the multimedia communication WebRTC use case is the one represented in figures below:

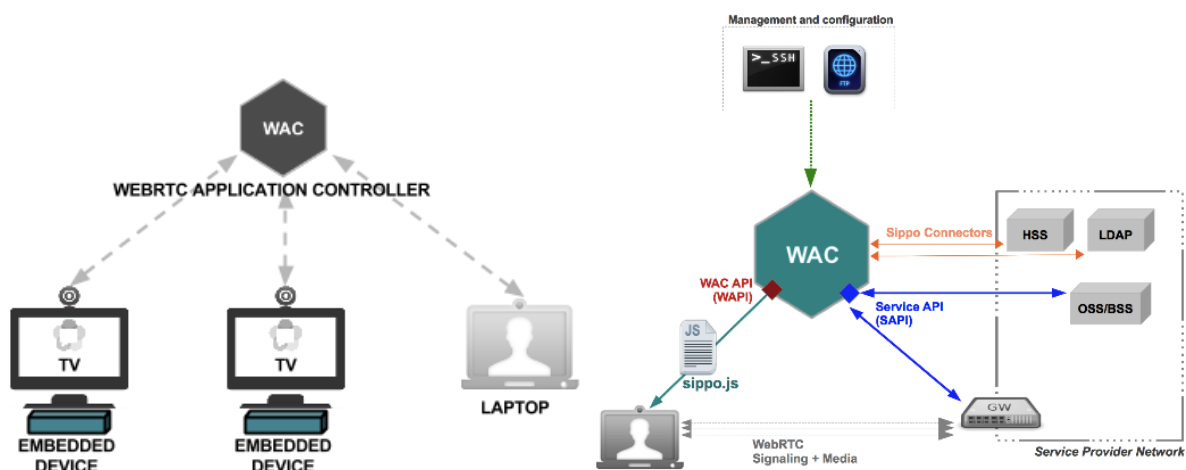


Figure 1: Demonstrator devices involved and Demonstrator architecture

Figure 1 represents the different devices involved in the demonstrator: a WebRTC Application Controller (WAC) and one or more groups of TV and Android TV device, and other devices like laptops, smartphones or tablets. The WAC and the interfaces showed implements the methods required to run WebRTC-based multimedia communications between the devices involved.

In order to show the results of the project, this demonstrator implements management of multicore in web environments.

Browser internals

Chromium

Chromium's multi process architecture, with different threads, each thread do different tasks. The GUI has its own thread, and I/O operation, for example.

Chromium has a process per tab known as “render processes” or “renderers” and it use “Blink” as layout engine. In general, tabs launch a new process, but if the number of process is too large, the process can be shared. Another exception is the use of `window.open` in Javascript, the window opened by this will share the process with the parent window.

How to see different process

Chromium has their own tool to see process, typing `shift + esc` in any Chromium tab it will open the task manager of google Chromium, or *Menu-> Tools -> Task Manager*. It provides a list of all the active process with the RAM, CPU and pid for each one.

Chromium runs each tab, window, plugin, webApps or extension in a separate process by default but there are some exceptions to this general behaviour:

Maximum number of process

When the number of process running is too high (is limited to 20 renderer processes in most of cases), tabs can shared resources, and reducing the load in the system.

Same domain

When the same domain is open in two tabs, these tabs can share the render process

Pages opened from Javascript

If a tab opens another one using Javascript it will share the resources to allow communication between two pages via JavaScript. This behaviour only occurs if the parameter `--process-per-site` is activated in Chromium. It can be activated:

- Windows: Open the properties of the Chromium launcher, and put `--process-per-site` in the *target* field and reinitiate Chromium.
- Linux: Launch Chromium with command line using *Chromium --process-per-site*.

Share process can be deactivated locally in a particular website, and can be forced to run in two separate process.

Forcing new pages to use their own process

It is possible to force a link to open in a separate process from a web page, just adding `rel="noreferrer"` and `target="_blank"` as attributes to the `<a>` tag, and then point it at a URL on a different domain name. Like this: `Google` In this case, Google Chromium knows that the page will be opened in a new window, that no referrer information will be passed to the new page, and that the `window.opener` value will be null in the new page. As a result, the two pages cannot script each other, so Chromium can load them in separate processes. Google Chromium will still keep same-site pages in the same process, to allow them to share caches and minimize overhead.

On the other hand, when pages use extensions, and multiple pages are loaded trying to use a certain type of extension, only one process it runs by extension, and the process is shared by the different pages.

Application level. Javascript

Javascript is a single-threaded language, and the ability to parallelize tasks at the application level is given by the workers. Workers were designed to take advantage of the multicore capabilities of the lower layers.

Workers

Workers provide parallelism in Javascript by creating threads at the SO level. There is no access to the DOM or any external component to the worker, so it is necessary a communication mechanism to send information between them.

The information exchanged by workers is sent by value, so each party works with a copy of the information. This way, there is no need to handle concurrency issues, since the memory is not shared. Workers have event `onMessage` to communicate, launched by `postMessage`. It is possible to use `addEventListener()`, in that case it's necessary to use the method `start()` in the port of web worker.

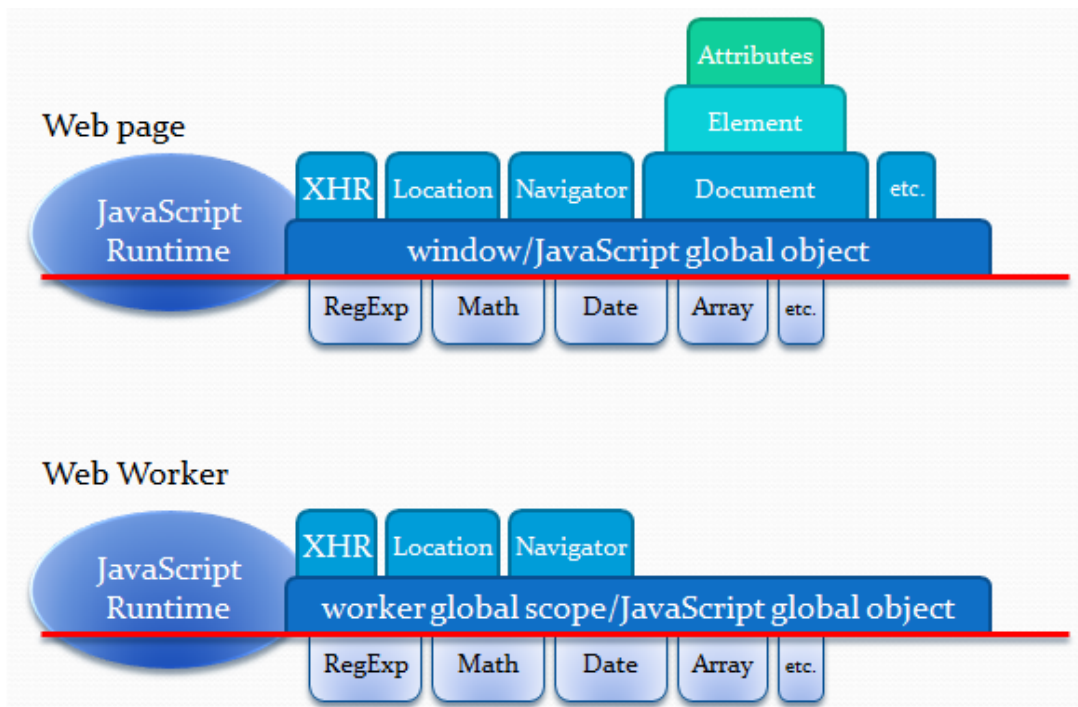


Figure 2: Separate DOM between page and worker.

Normally, the workers have a long life cycle. It has high start-up performance cost, and a high per-instance memory cost. This is why the runtime will be longer than the init time.

Workers need the URL or path of a Javascript file in the constructor, and return an object worker. Workers can be shared between different pages, whether they are independent pages or in an iframe.

Web workers are used in this demonstrator for:

- Processing image or video: For example split a `<canvas>` in several zones and different workers work in parallel in the imagen.
- Parse data from `XMLHttpRequest` call if the required time is high.
- Concurrent request to local database. Workers allow thread safe request.

Web Components

Web components are one of the main enhancements in latest versions of HTML. They are intended to build reusable components and widgets that can be used in a transparent way. These components add features and functionalities like:

- Custom elements. It allow to define custom objects.

- HTML templates
- Shadow DOM to generate a separate DOM from main page.
- HTML imports

Web components are not fully implemented in all browsers yet. The next picture shows the current state.

	Specced	Implementation				
		Polyfill	Chrome / Opera	Firefox	Safari	Edge
Templates			Stable	Stable	8	13
HTML Imports			Stable	On Hold	No Active Development	Vote
Custom Elements			Stable	Flag	Prototype	Vote
Shadow DOM			Stable	Flag	10	Vote

Figure 3: State of web components

2.3 Results

To check the performance and viability of the proposed architecture including the management of different media sources over a multicore architecture, some tests have been done. The results are good and the time required for specific critical operations decrease using multicore. But there was a case where the result is not valid for this use case. It was the file transfer. It was no possible to manage this transfer since the browser managed it in their own thread without using workers.

Details of the results are available in the appendix 9.

2.4 Conclusions

The result of this use case is a demonstrator of unified communication services using HTML5 based web browsers on embedded systems. Media sources have been distributed in different processes obtaining some improvements in terms of priority management, bandwidth use and timing. Although it is worth mentioning that some problems have been found for file transfer use cases due to the browser capabilities. The resolution of these incidents is not the scope of this project, but its resolution in the future would enable the architecture implemented in this task for a wide set of use cases.

3. USE CASE OPEN DETERMINISTIC NETWORKS

3.1 Use case introduction

The use-case demonstrates a novel approach for monitoring people on airports and measuring passenger data by interconnecting cameras, servers and clients. The person tracking application, which creates metadata by extracting semantic information from image data and the additional middleware, both being developed in WP12, are used to implement the demonstrator's passenger monitoring. Cameras (servers) and clients are specifically connected via Ethernet to stream video and TTEthernet for exchanging generated metadata. TTEthernet is based on a deterministic communication architecture which is standard for hard real-time systems for example in aircrafts and vehicles. The specific system behaviour enables two applications:

1. Cameras (servers) and clients are specifically connected via TTEthernet for ensuring that the transmission of metadata (e.g. alerts) does not fail even during a network congestion (Figure 4). Such latency is a serious problem when seeing camera networks as part of future cyber-physical airport systems, for example the rapidly emerging demand of measuring passenger waiting times in real-time at security gates for automated proactive intervention into the gate process.

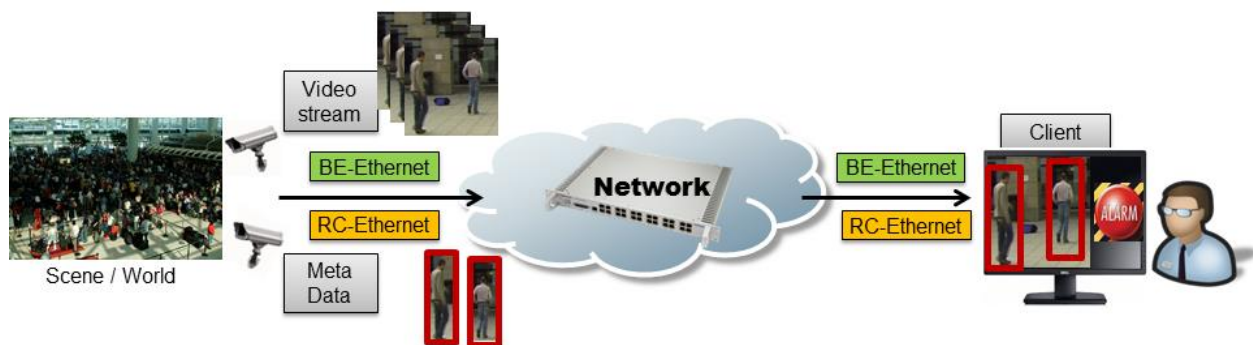


Figure 4: Alerting staffs in case of an incident (e.g. amount of people exceeds threshold) by utilizing a deterministic network of cameras, servers and clients

2. Reliably tracking a crowd of passengers can only be achieved with distributed smart cameras, which perform computer vision tasks using distributed algorithms. Distributed algorithms scale better to large networks of cameras than do centralized algorithms. Limited network bandwidth and the non-deterministic behaviour of Ethernet do not guarantee that information between different sensor units (distributed algorithms) can be shared on time as requested by distributed object tracking algorithms. A deterministic network (TTEthernet), however, overcomes these problems and enables the use of distributed algorithms in visual sensor networks.

Adjacent to the above demonstrator, DTU has developed for Deterministic Ethernet networks a design optimization tool named DOTTS that determines the routing of frames in the network, the schedule of the Time-Triggered (TT) frames and the packing of messages in frames such that all frames are schedulable and the worst-case end-to-end delay of the Rate-Constrained (RC) messages is minimized. This demonstrator will focus on the next evolution of Deterministic Ethernet networking technology in the scenario of the Internet of Things. The main goal is to demonstrate the possibility to integrate traffic streams of mixed-criticality in one physical

network. DTU developed an optimization tool for incremental scheduling of Deterministic Ethernet networks, which allows adding new TT traffic with no or minimal impact on existing traffic, thus minimizing the revalidation and recertification costs of the system.

3.2 Description of final demonstrator

The final demonstrator consists of two PCs (Linux Ubuntu 14.04 LTS) equipped with TTEthernet capable PCI Ethernet cards connected via Ethernet cables to the industrial Gigabit switch - TTESwitchChronos 18/6 Rugged¹ (Chronos). This switch supports time-triggered (accord to SAE AS6802), rate-constrained and standard Ethernet (IEEE 802.3, IEEE 802.1Q) traffic flows. The two processing units, called sender and receiver, are used to transmit video sequences and metadata over the network.

The demonstrator consists of two applications:

1. Streaming video data and transmitting metadata
2. Distributed computer vision algorithms - passenger tracking

Streaming video data and transmitting metadata

The network interface cards and the Chronos switch are operating in two modes of communication. The first is a best-effort communication which is the common mode of operation in Ethernet. On both sender and receiver a daemon is running, that allows the NICs to be operated exactly as any other network card using TCP/IP. However, limited network bandwidth and the non-deterministic behaviour of Ethernet do not guarantee that alarm data arrives on time at the receiver side.

Therefore, the NICs and the Chronos switch are configured to provide a rate-constrained (RC) communication channel for alarm data. The RC-channel is provided by the TTEthernet framework of the communication devices.

The application consists of two programs:

- one for sending a video stream and its alarm data which is running on the sender
- one for receiving a video stream and its alarm data which is running on the receiver

Streaming is done by using FFMpeg² utility. The transmission is done in a direct RTP stream to a multicast address that the receiver has subscribed to. The libraries grab the current frame and encode them into MJPEG format chunks which are then packetized and transmitted. The receiver application is based on the FFPlay utility. It is configured via a Session Description Protocol (SDP) file and collects all RTP packets that arrive at the given multicast address. Packets are reassembled and frames are decoded.

The applications also provide the means to operate the rate-constrained communications channel. Network congestion causes the RTP stream to jitter and fail as not all packets will arrive at the receiver in time. The alarm, however, is transferred in a deterministic manner and is therefore always readily available at the receiver. In case of lost frames, the last received frame is displayed but the correct metadata (alarm/ no alarm) is written in the overlay.

¹<https://www.tttech.com/products/industrial/switches/rugged-switches/tte-switch-chronos-186-rugged>

² see www.ffmpeg.org

Distributed computer vision algorithm - passenger tracking

For many crowded scenarios, passengers are heavily occluded, especially from a certain viewing point of a single camera. Another camera which is placed for a different viewing point can often solve occlusions for the same scenario. So it is necessary to take consideration of the metadata from another server (camera) which encodes a flag when an alarm condition is raised from the other camera. The extracted metadata needs to be reliably exchanged near real-time. Therefore, the NICs and the Chronos switch are configured to provide a rate-constrained (RC) communication channel to exchange metadata between both units. This mode enables a highly efficient and robust communication, so that for the same moment of the scenario, various visual sensors are watching the same scene and helping each other by exchanging relevant information. Details about the application are summarized in appendix 10.

Quality of Service Networking (DTU)

For this purpose, the demonstrator has been built up using newly developed hardware components by TTT and experiment with the design and configuration tooling that is subject to the development in WP3.

To configure a TSN network, two challenging tasks that have to be solved are the routing of streams and the schedules of TT flows. Within this explicit work, real-time applications implemented using TSN distributed cyber-physical systems are considered. As an input to the defined problems, the following is used as input: (i) the network topology, (ii) the set of TT flows and (iii) the set of Audio Video Bridging (AVB) flows. We are interested in determining the TT schedules called Gate Control List (CGL) and the routing of the TT and AVB flows, such that all flows are schedulable and their WCDs of AVB flows is minimized.

3.3 Results

Streaming video data and transmitting metadata

The sender transfers the frame number and the according alarm state. The receiver collects the alarm data and displays the received frames with an overlay for alarm information (Figure 5).



Figure 5: An alert is thrown if a person enters the lawn (right image).

Simulating effect of network load on packet loss

A noise generator is connected to the network as well in order to simulate network overload. In such situations the demonstrator shows that event data in rate-constrained mode arrives correctly

at the client whereas video directly streamed to the client under best-effort mode shows frame drops. The number of frame drops strongly depend on the network load. Our simulations verify that deterministic network communication allows safe processing of video and safe event delivery at the client side which is a necessary prerequisite when putting cameras into the control loop of the gate process.

Distributed computer vision algorithm - passenger tracking

Due to occlusion, the receiver camera has difficulty to detect the correct amount of passengers. In the example (Figure 6) three passengers are detected as one in Camera 2:

- a) One person is behind the front person, which results to loss detection.
- b) Two persons are geometrically close, which results in a merge of detections.

Camera 1, however, correctly detects 4 separate passengers, which has a better understanding of the current scene. These bounding boxes with geometric information are sent to camera 2, which may resolve the occlusion and improve its tracking.

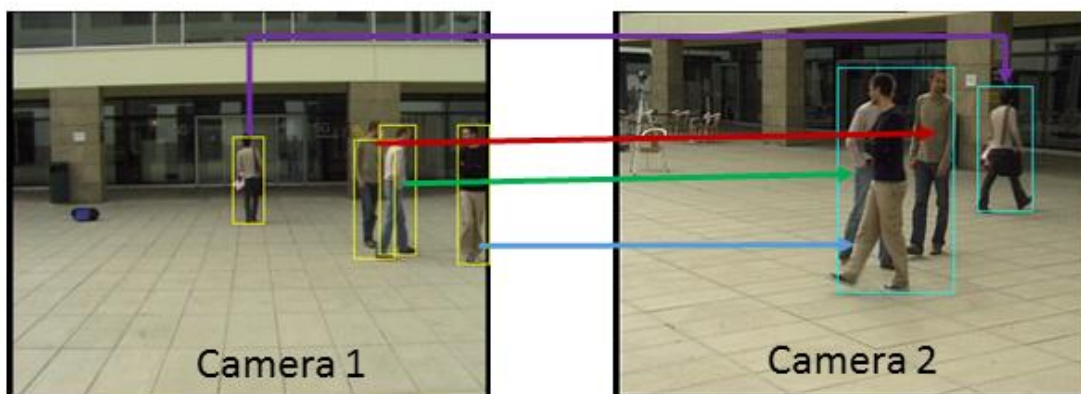


Figure 6: Camera 1 (sender) detects 4 passengers. Camera 2 (receiver) by itself detects only two passengers due to occlusion. Camera 2 receives the information that camera 1 detected 4 persons, which helps to resolve the occlusion.

Quality of service networking (DTU)

For the CGL synthesis, TTT has developed a Satisfiability Modulo Theories (SMT) approach to find a solutions which satisfies an imposed set of constraints³ and DTU has proposed Integer Linear Programming (ILP) formulation⁴. For TT flow routing, TTT has developed a simple heuristic that minimizes bandwidth. However, for routing AVB flows, we have jointly proposed a Greedy Randomized Adaptive Search Procedure (GRASP) –based heuristic. These methods have been developed as offline tools; see Figures 7 and 8 for an illustration.

³ S.S. Craciunas, R. Serna Oliver, and M. Chmelik, “Scheduling real-time communication in IEEE 802.1Qbv Time Sensitive Networks”. In *Proc. Of International Conference on Real-Time Networks and Systems*, 2016

⁴ P. Pop, M.L. Raagaard, S.S. Craciunas and W. Steiner, “Design optimization of cyber-physical distributed systems using IEEE time-sensitive networks”, in *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 86-94, 12 2016

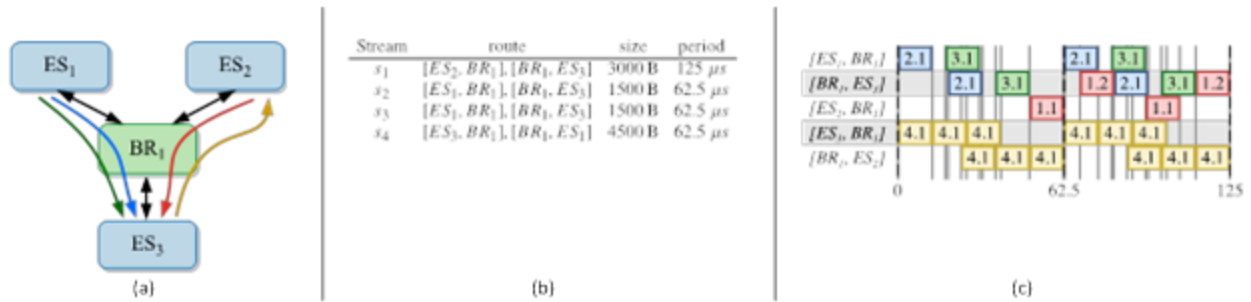


Figure 7: Schedule synthesis example – (a) Network topology, (b) TT flows, (c) Feasible scheduling solution

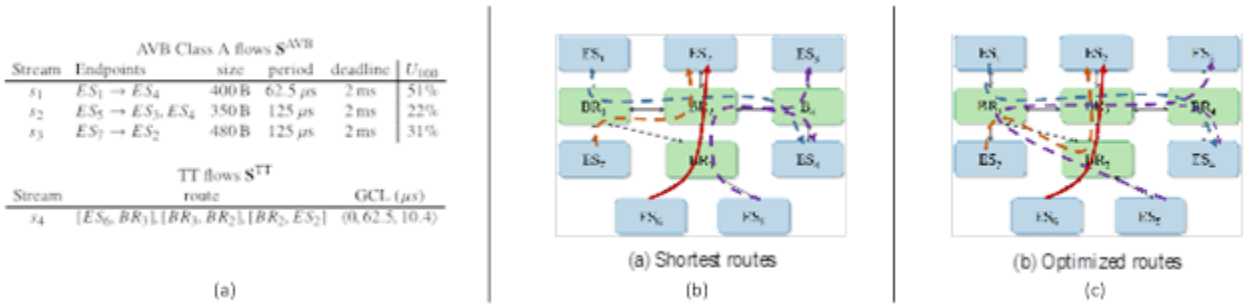


Figure 8: Routing synthesis example – (a) TT and AVB flows, (b) Routing solutions, (c) Optimized routes

However, open adaptive networks require reconfiguration at runtime. Therefore, we have also developed heuristic algorithms for the configuration of TSN networks which can quickly determine at runtime a good quality solution. For the CGL synthesis, we have proposed an As Soon As Possible (ASAP) heuristic and a GRASP-based heuristic⁵, and for the routing we use Graph-based algorithms to find the shortest paths. The proposed ASAP and GRASP heuristics have been evaluated against the ILP solution, showing that it can quickly find good quality solutions, hence they can be used for runtime reconfiguration in open adaptive networks.

3.4 Conclusions

This use-case presents two computer vision demonstrators based on deterministic networks that tackle the challenge of passenger monitoring on airports where the dynamics of the network load over time, pose significant problems to the design of a surveillance system as well as its distributed computer vision algorithms. The proposed demonstrator solves the problems by 1) transmitting metadata over the deterministic network (RC-Link) while the video is streamed over BE-Ethernet, 2) extracted Meta data can be shared near real-time without delay or packet loss over the RC-Link.

Given a specific network load, the presented demonstrator is able to transmit Meta data over the RC-Link while video packets are lost. Thereby, the relevant information (such as alarms) are arriving on time at the control station even if the network is congested. This system behaviour is of great importance for safety and security related applications; e.g. passenger tracking for surveillance. Experiments confirm the hypotheses that metadata still arrives on time without any

⁵ Michael Lander Raagaard, *Algorithms for the Optimization of Safety-Critical Networks*. Master’s Thesis, Technical University of Denmark, January 2017

significant delay even if the network load is severely increased. Furthermore, distributed computer vision algorithms can greatly benefit from deterministic networks, because their characteristics first utilize those solutions.

4. USE CASE AUTONOMIC HOME NETWORKING

4.1 Use case introduction

The Autonomic home networking use case aims to deal with numerous multi – criticality issues in an autonomic smart – building environment, through the design and development of a decentralized system consisting of autonomic distributed decision making units, which can be connected to and control multiple building devices. The decisions of these units may lead to a set of “solutions” $s_i \in S = \{s_1, s_2, \dots, s_N\}$ where each s_i corresponds to a different combination of the home’s devices operations.

The envisioned system will be able to manage multiple data flows of information and requirements, with different weights of importance, and come up with the best solution of this solution set. The main requirements of the system may be categorized as follows: (i) Building’s total energy consumption minimization, (ii) Services’ delivery delay minimization and (iii) Users’ satisfaction level.

The planned demonstrator / prototype, described in detail in the next section, will be based on an autonomous smart building system. Data will be gathered from a set of sensors embedded inside the house, regarding both the environment and the state of the user. The data will be available through a registry unit, and will be accessed in a service – based basis. Each time the system is triggered by a service, data will be analyzed in a distributed manner, and the system will decide upon how devices in the house should operate in order to meet the user’s needs, meaning that the system will choose from one of the N possible solutions.

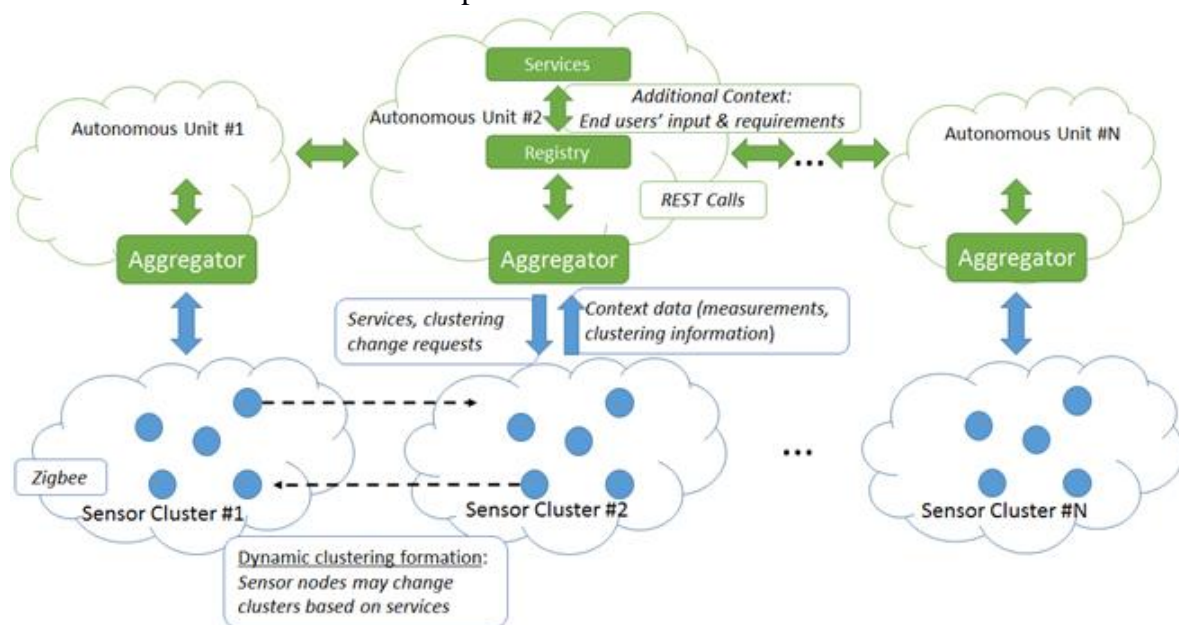


Figure 9: The Autonomic home use case overall design

4.2 Description of final demonstrator

The final demonstrator will be based on a SoA architecture, where each interaction with the WSN world, can be modelled as a Service that is provided by an entity that can control the WSNs. As such, a user, which may be a human or another automated system, that needs to interact with a sensor for e.g. get the temperature of a room or the current RPMs of a motor, can call the corresponding service and get the result of the sensor through the entity that is responsible for contacting this sensor.

In more detail, the demonstrator's key elements are the aggregator units. The aggregator units are the entities that have specialized hardware to talk with the WSNs and will provide to the rest of the system the services for controlling the WSNs. In the architecture that will be demonstrated, two heterogeneous WSNs will be presented: their heterogeneity comes from the protocols used to communicate. The sensors create clusters, and each aggregator unit is responsible for one cluster. These clusters are formed dynamically, based on location of the nodes as well as of the types of data they can collect. Upon request (e.g. from a specific service), modifications may be required and the clusters may be re-formed. The communication within this clusters is based on specific protocols compatible with the sensor devices (e.g. Zigbee).

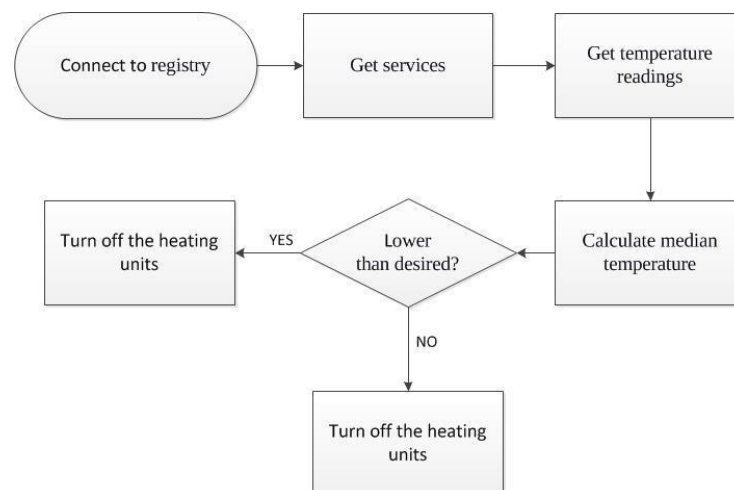


Figure 10: Demonstrator

The demonstrator will also include the registry unit: The registry unit acts as a service discovery entity, where aggregators are required to register their services to, as well as update and delete any service needed, and users can poll the registry unit for information about services or aggregators. In the demonstrator aggregators will communicate with each other, and thus may use the registry unit as users, in order to get information. Finally, in order to cater for the multi-criticality aspect, the service requests come with some extra contextual parameters. Each call to a service is accompanied by a criticality level that the user would like this request to be treated with. The criticality level defines how important a service call is, and will be discussed in detail later in the paper. By using the criticality level of each request, the aggregator unit is able to treat each request differently. The overall aim of the aggregator unit is to provide services to users to a certain level of quality, while also taking into consideration policies defined beforehand, i.e. minimizing energy consumption.

The use case to be demonstrated will be based on temperature setting: the service offers the ability to control the heating of a specific location. The user can set a desired temperature and the client will ensure it will be applied, with the aid of one or more temperature sensors and one or more actuators that control heating units at the premises.

The specific service is not assigned high criticality, since the changes in temperature do not take place suddenly but instead gradually. Moreover, the late response in a temperature change does not lead to irreversible effects; hence the chosen criticality is medium.

4.3 Results

The expected results in the demonstrator are related (i) to delays regarding the completion of specific services; (ii) the level of criticality assigned to each service requested and (iii) the estimated energy savings due to the decisions made throughout the service request meeting procedure.

4.4 Conclusions

This use case aimed to create an autonomic smart – building environment through the design and development of a de-centralized system consisting of autonomic distributed decision making units. The implemented system included a variety sensing units / technologies comprising a variety of different sensors, provided by all three partners of this task (HUA, AMBAR and IMA) structured in to complementary systems. Both systems include fully – implemented multicore functionalities, in order to be able to support multi – critical services related to smart building environments. In regards to the IMA multi-core multi-critical wireless gateway, a quad-core ARM SMP embedded device was used as a HW basis. The device has 2 high speed USB2.0 host ports for attaching wireless devices and a 10/100Mbps Ethernet with RJ-45 LAN jack for wired communication. The gateway was powered by Linux OS with custom modifications. Multi-criticality was achieved on SW level. The second sensing unit was the one of Micaz, where a raspberry unit has been used as an aggregator to achieve multi – core functionality.

Overall, a diverse set of hardware components was integrated, and innovative software components were developed, towards the final system that is demonstrated. The system is able to tackle with multiple services with different criticality levels: the services are allocated to different cores based on these levels. The time delays, the number of different service requests as well as the estimated energy savings are the key measures and indicators used in order to evaluate the system's efficiency.

5. USE CASE ULTRALOWPOWER HIGH DATARATE COMMUNICATION

5.1 Use case introduction

The ultimate goal of this ultra-low power high data rate project is to realize a high data rate wireless SOC [System On Silicon] which combines re-configurability with very ambitious performance goals.

The demonstrator realizes both a BTLE [Bluetooth Low Energy] and 802.15.4 functionality demonstrating the architecture is able to support multiple standards by modifying the software. The architecture will allow also for evolution into a WLAN [Wireless Local Area Network] SOC.

The demonstrator exhibits breakthrough performance in terms of power consumption and link budget. Link budget defines the distance over which a radio link can communicate.

In the EMC² project the solution is demonstrated on an FPGA based platform.

Mixed criticality challenges to the multi-processor architecture:

Wireless high data rate networks typically consist of a number of communication protocol layers and processes which present conflicting requirements to a multi-processor architecture.

At the lowest level [physical layer or PHY] data are extracted by demodulation. This process is very compute intensive and has strict latency requirements [the sheer extent of the incoming data limits buffering capabilities]. Architectural options are also limited by power consumption considerations. The desirability of realizing this on a processor platform depends on the used modulation and the required data-rates. Although the BLUSP processor platform is well suited to handle complex modulation [e.g. OFDM] at medium data rates, for BTLE [GFSK] and 802.15.4 [QPSK] the choice was made to go for a dedicated data path based architecture.

At the networking layer level low latency or fast response time and high throughput or limited amount of processing are driving the architecture. The protocols implemented in this demonstrator – Bluetooth low energy, 802.15.4 and [future] WLAN – support the notion of connected links which require a solution to process packets and deliver packets on the air within strict time limits. Latency requirements present itself under different forms: [1] timely transmit of receive acknowledge information, [2] requirement for fast packet source check and response, [3] at a higher level use cases as e.g. streaming audio data transfer which require low guaranteed latency and high throughput. And these low latency and high throughputs have to be achieved under very strict power consumption conditions.

Whereas the PHY and the networking layer are standardized, the application layer allows for a large variety of use case specific functionality. The challenge at this level is to allow for ease of implementation, flexibility without interfering with the critical processes of the lower layers.

5.2 Description of final demonstrator

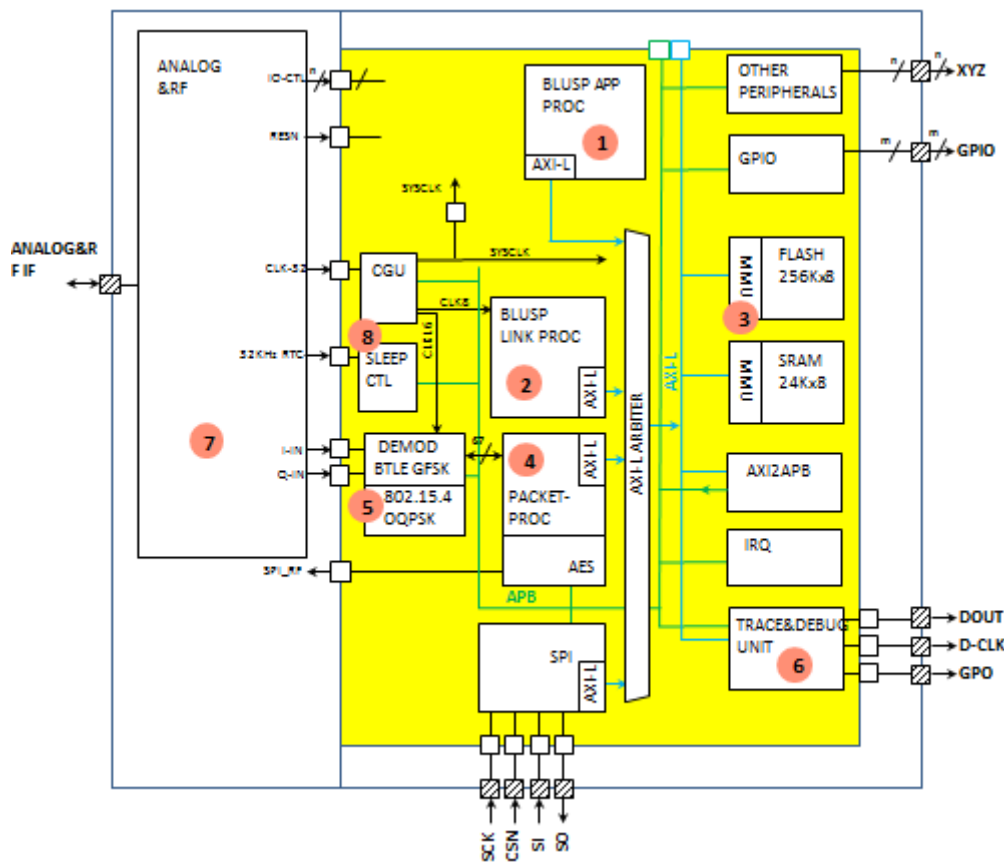


Figure 11: Hardware architecture

The hardware architecture consists of a twin processor system [see figure 12], a memory subsystem and a number of [hardware] co-processor blocks.

PHY

In the system design phase the conclusion was reached that for the modulation used in BTLE and 802.15.4 [GFSK and QPSK] the most efficient implementation is a dedicated micro-programmable data path.

TWIN processor system

The complexity of multiple conflicting critical processes has been solved by combining:

- () a hardware packet processor which [4 in Figure12] transmits and receives the packet from the wireless link from and into the retention memory [3 in Figure12].
- () a link processor [2 in Figure12] which configures from time to time the packet processor, manages the control procedures and the link security. Many of these processes require strict processing delay control. In a separate thread other less timing constrained processes are running to manage the connections, observe links etc.
- () a host processor [1 in Figure12] which manages application processes which can be simple for e.g. a temperature sensor node to very complex for router nodes, or edge nodes. Communication between both processors is managed by processes in separate threads.

() a MMU [memory management unit] which allows to split the memory into retention and no retention memory, allows to share or keep private part of the memory between the 2 processors and allows to have secure memory parts for security procedures.

Memories

The applicability of ultra-low power memories as developed in WP2.2.3 is an architectural improvement of interest. This technology is relevant for a wireless multi-radio demonstrator, of which power consumption both in active and inactive – sleep mode is critical. Power consumption especially in inactive mode is determined to a large extent by the data memories. This will become more critical into the future, by the increasing protocol complexity, more data bandwidth, more demanding use cases. The fact these memories are less reliable can be managed by introducing sufficient error correction and detection.

BLUSP DSC processor

A short description of the BLUSP processor core. The BLUSP is key in the low power performance of presented solution: its high performance allows to meet the strict latency and throughput requirements even keeping the clock frequency very low.

BLUSP is a 3 issue core: it executes 3 instructions in parallel. It is able to execute one complex multiply accumulate [MAC] in parallel with one ALU instruction in parallel with one load store instruction. Each of these instructions are executed on a separate execution [EXU] unit.

It is a ULTRA RISK architecture with an instruction set of about 90 instructions. All instructions execute in single-cycle.

Its DSP capabilities include support of fractional fixed point data-types, saturated add/multiply and native instructions operating on complex fractional data types. Its complex multiplication/accumulation unit supports single-cycle 16x16bit complex multiplication and addition. Its register file comprises 32bit registers.

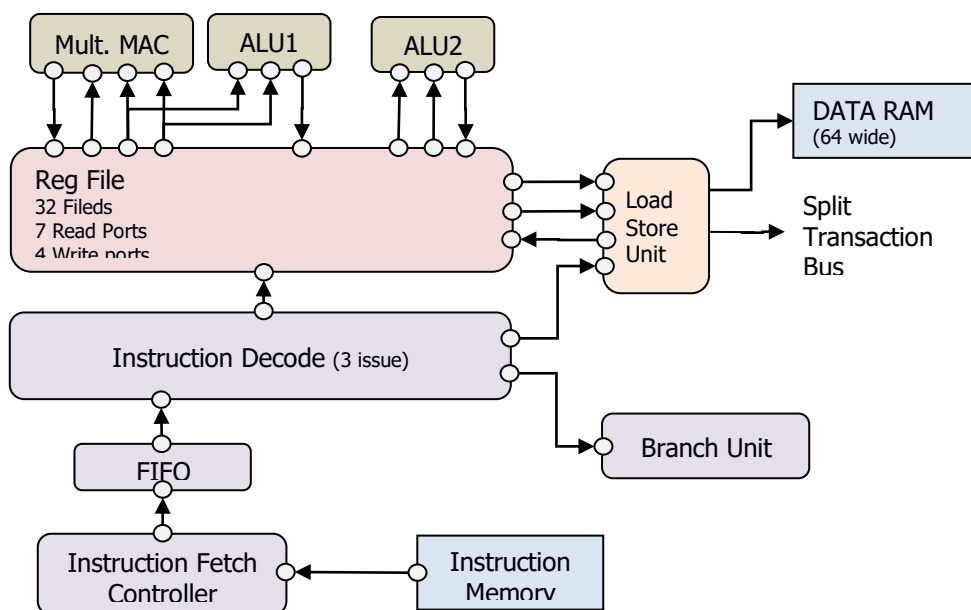


Figure 12: BLUSP architecture

Its instruction word is 64bits long, containing 3 instructions which can or cannot be executed in parallel. It supports 64-bit data memory. 8-bit, 16-bit, 32-bit and 64-bit load/store instructions are part of the instruction set.

Its programming model is very straightforward. It can be fully programmed in standard C. To use its complete DSP capability, one specific C data type has to be used: “Complex Fract Short Int”.

Software

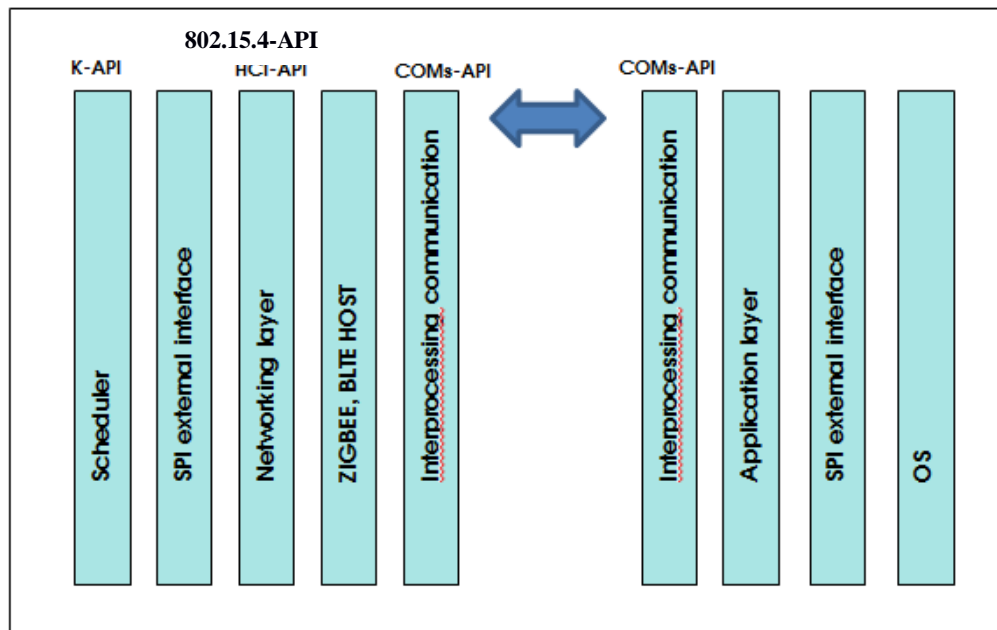


Figure 13: Software architecture

The software consists of a number of threads which are running on the 2 processors, each managing processes with a number of specific performance requirements. Not all the software threads have been implemented inside this project. On the LINK processor [2 in figure 12] a scheduler is managing the processor item allocation and the priorities of each of these processes. This scheduler is interfacing to a K-API [Kernel API]. On the APP processor [1 in figure 12] an OS will be implemented.

This OS remains to be selected. An appropriate wireless sensor operating system. Operating system alternatives are RIOT, Contiki, TinyOS, Linux... The RIOT operating system is a promising recent open source European initiative focused on embedded nodes networks that requires low power and computational resources.

Short description of the software components: at networking layer: BTLE Link layer interfacing to an HCI –API. HCI is also accessible as HCI commands on the SPI interface. The BTLE link layer is Bluetooth specification V4.0, V4.1 and V4.2 compliant. 802.15.4 MAC layer interfacing to a proprietary 802.15.4-API. Also accessible on the platform SPI interface. The MAC layer is compliant with the 802.15.4 standard. The ZIGBEE and BTLE thread are based on external solutions.

SPI external interface. SPI-based debugger that can survive the deep sleep mode. Problem with JTAG debugger is that they require the JTAG interface to be powered in order to be able to run the debugger. This is an issue in this device, as the goal is to put the device in a sleep state most of the time where the JTAG is not powered. The issue can be resolved by doing a SPI debugger. Application layer on the application processor. Communicates with the LINK processor over a COMs API.

Following table summarizes the software memory [program and data] sizes.

Module	Status	Program memory
Link Layer	Developed	16 KBYTE
802.15.4 MAC	50% developed	16KBYTE
SCHEDULER	Developed	15 KBYTE
BTLE HOST	External	38KBYTE
BTLE Profiles	External	Partner
ZIGBEE	External	55KBYTE

Table 1: Software component overview

5.3 Results

FPGA environment - Overall system performance

The ultra-low power high data rate communication module has been realized on a ZYNQ FPGA board [figure 15] interfacing to a 65nm RFCMOS RF frontend. This RF chip has been developed by a BlueICe partner company outside the scope of this project.

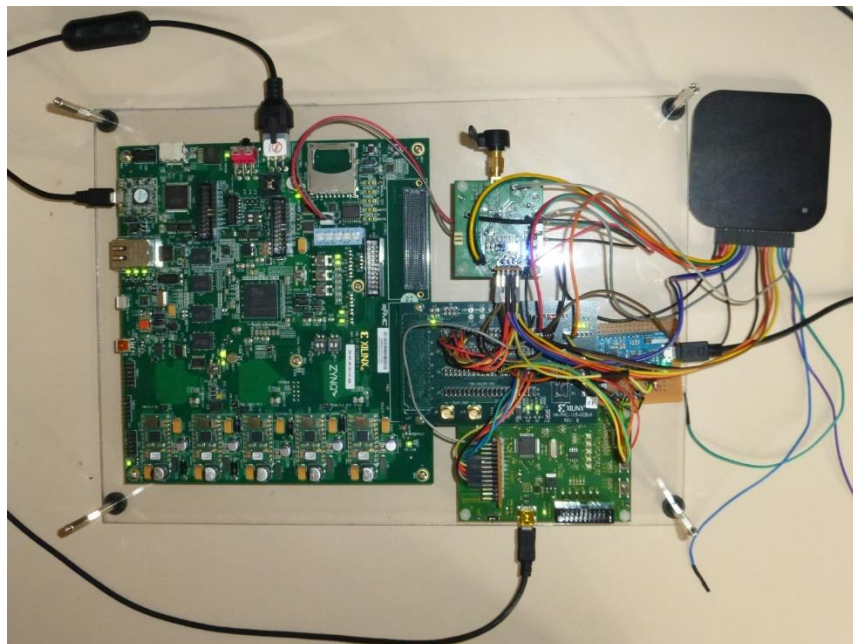


Figure 14: FPGA platform

The complete demonstrator [figure 16] consists of [1] two of these FPGA boards, [2] a number of commercial BLTE and ZIGBEE nodes, [3] a PC based scenario engine, and [4] a tablet which interfaces to the wireless network.

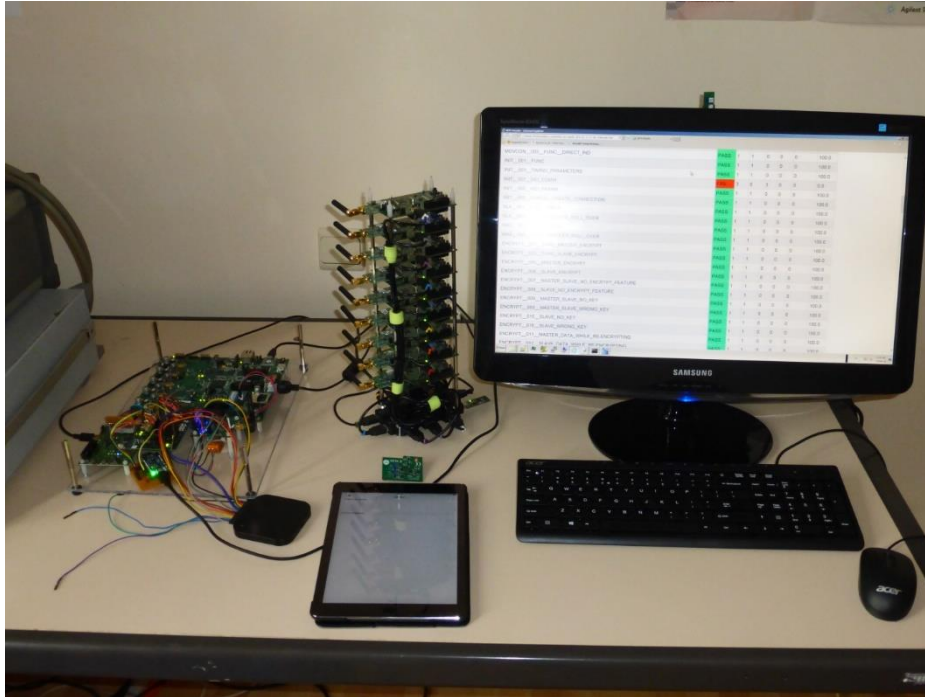


Figure 15: Complete solution

The PC based scenario engine, launches a number of link layer scenario's to the Device Under Test [DUT] and reference devices through the Test board. This is used to demonstrate the high data rate communication module and is also used to do thorough validation testing.

Test scenario group	# scenario's	Status
BTLE link layer slave	100	All pass
BTLE link layer master	100	All pass
BTLE link layer multi-master	50	All pass
802.15.4 basic	50	All pass
802.15.4 complete	50	All pass

Table 2: validation scenario overview

Breakthrough linkbudget

The demodulator [5 in figure 12] performance has been measured with an RF test chip [external to this project]. Results show great sensitivity and power consumption. In figure 17 sensitivities

are given, compared to present market available solutions. Very significant improvement over the state of the art has been shown.

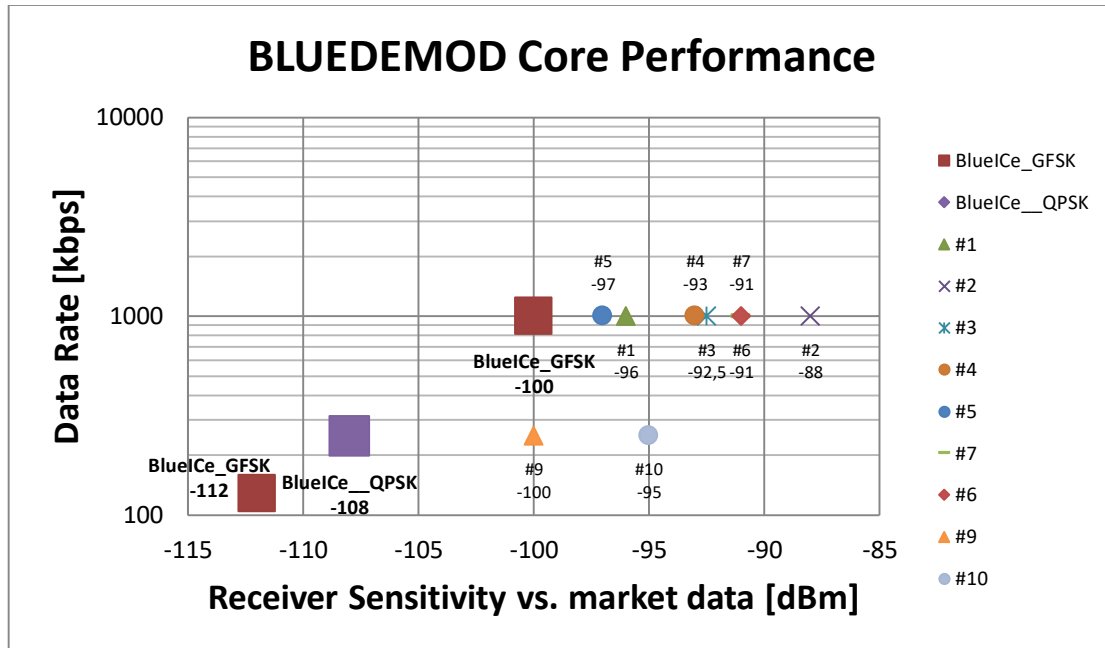


Figure 16: Ultra low power high data rate communication receive sensitivity

Breakthrough power consumption

The PHY power consumption/performance combination results are outstanding. This has been also achieved through advanced RTL design methods. In table 3 a summary of the demodulator power consumption is its most important modes.

Scenarios	Area (um ²)	Total power (uA)				
		Correlate	Decode	Active	Idle	Packet
Manual Clock Gating	51655	480	361	380	11	305

Table 3: GFSK demodulator power consumption [90nm TSMC reference process]

The packet processor platform power consumption is also outstanding. This is due to [1] its architecture which only keeps the LINK processor active for a short time and typically not when the radio is active, [2] the very low power consumption for the BLUSP processor and [3] the BLUSP performance which allows to keep the clock frequency very low.

To demonstrate this the power consumption for a BTLE RX – TX combination is calculated and compared to 2 industry examples.

	410μS			150μS			410μS			150μS		
"on air"	RX			Tifs			TX			RX		
Current consumption[1]	[Red Bar]			[Red Bar]			[Red Bar]			[Red Bar]		

	[Redacted]											
	10µA	350µA	1mA	10µA	10µA	90µA	1mA	10µA	10µA	350µA	1mA	10µA
	100µS	310µS	95µS	55µS	100µS	310µS	95µS	55µS	100µS	310µS	95µS	55µS
Phase	I	II=mode1	III=mode2	IV								
average digital current, receive packet	565µA											

Figure 17: Digital current profile

This current profile is based on RTL simulation for a TSMC 90nm CMOS reference process. The total digital energy for a 32Byte BTLE packet equals 0.2 µJ [RX] and 0.12 µJ [TX] = 0.32 µJ for an RX-TX.

Compared this to 2 typical commercial solution [Figure 19]: solution-1: [65 µJ RX-TX]; solution-2: [15 µJ RX-TX]; these commercial solutions include also RF power consumption. The digital consumption developed solution consumes less than 2.5% of the commercial solutions. The commercial solutions include the RF current. In the 2 solutions the RF current is estimated at 75% of the overall current. So an improvement by a factor 20-80 has been achieved.

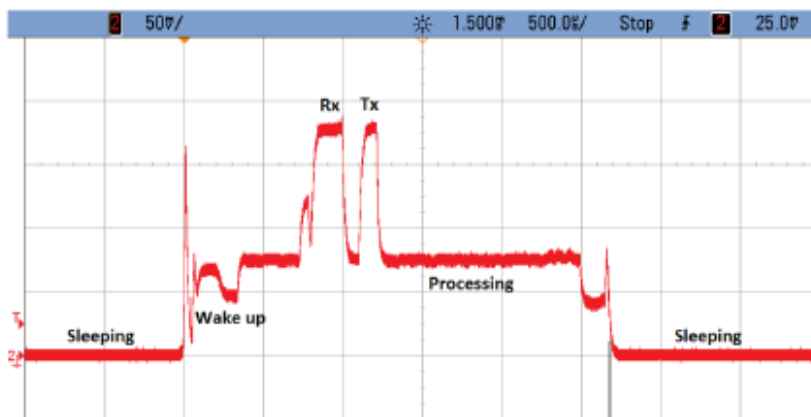


Figure 2- Current Consumption versus Time during a single Connection Event

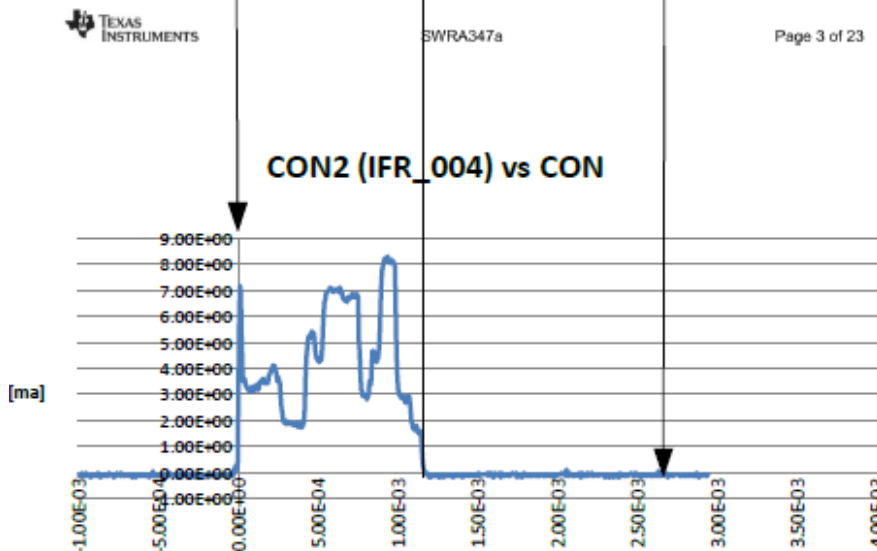


Figure 18: Commercial product current profiles

Reconfigurable solution and software development friendly

A software/hardware reconfigurable architecture has been shown which allows for significant performance.

The demonstrator confirmed the ease of software development and debug paradigm.

5.4 Conclusions

The realization of the demonstrator was a success:

- (1) An architecture has been developed which was demonstrated to result in a breakthrough link budget for a BTLE and 802.15.4 demodulator. During the last year BlueICe used this architecture for other wireless packet oriented communication standards as e.g. DVBS2X satellite communication. Also here the architecture confirmed its breakthrough link budget.
- (2) The second goal was to achieve breakthrough power consumption. Also this goal has been achieved, even if typically processor based architectures tend to suffer on this solution parameter. Some highlights:
 - a. (1) a power consumption of less than 300 μ A has been shown for the PHY part of the platform [the demodulator]. This in in a reference 55nm TSMC technology.
 - b. (2) sleep power of less than 0.5 μ A has been demonstrated. This is possible by the fact in SLEEP mode only a small fraction of the platform SRAM needs to be supplied. Power supply of all other parts of the system can be removed.
 - c. (3) active processor power of the BLUSP core is ~50% of comparable existing architectures. Moreover the performance of the processor allows it to be operated at a very low frequency.
 - d. (4) the overall solution has been architected in a way the processor system in only active when the radio link is not active, reducing peak power numbers.
 - e. (5) the overall solution has been architected in such that the processor system which is the most power hungry part is only active for a short time periods.
- (3) The third ambition was not to increase complexity expressed in area needed for an SOC implementation too much. The goal was not to be bigger than 120% of a full hardwired solution.
- (4) Work has been done on exploitation of the results. Some example actions/achievements
 - a. DEMOD has been licensed to an alpha customer. This customer integrated the demodulator with his RF front-end in a 55nm CMOS process.
 - b. BLUSP has been packaged as a commercial licensable IP core. Significant interest has been detected in the market.
 - c. PHY architecture has been reused for a DVBS2X implementation.
 - d. Architecture has been reused for a combo Bluetooth Classic, BTLE solution.

6. USE CASE SYNCHRONIZED LOW-LATENCY DETERMINISTIC NETWORKS

6.1 Use case introduction

As previously stated on other WP11 deliverables, this demonstrator aims to increase the dependability on communication networks for Smart Grid, focusing on availability, safety, security and reliability. Control, monitoring and critical data rely on these features and thus, on synchronization. Timing data becomes crucial since it is required to provide properly the interconnection between different grid elements and guarantee their functionalities. Next Smart Grid network generation rely on high-accurate, reliable, available and scalable technologies to distribute time. White Rabbit (WR) is intended to be the next-generation deterministic network based on synchronous Ethernet, allowing low-latency deterministic packet routing and high precision for timing transmission. WR networks are composed by a master node that provides the main time and frequency reference to fibre switches, twisted-pair copper, and slave nodes. WR allows a very precise time-tag data measurement and data triggering acquisition in large installations at the same time that data are also transmitted through the same network.

This demonstrator addresses the development of a novel technology for Smart Grid control based covering the following features:

- Deterministic high-accuracy time and frequency transfer
- High-availability of network services
- Fault tolerance and robustness
- Heterogeneous synchronization mechanisms support
- Scalability
- Safety and security (integration of Telvent (Schneider Electric) WP6 results)

6.2 Description of final demonstrator

During the second year of the project, we shaped the final design of this demonstrator with a reasonable number of elements to cover all the stated features and functionalities of the use case. Next Figure represents the final design of the demonstrator that was formulated during the second period of this project.

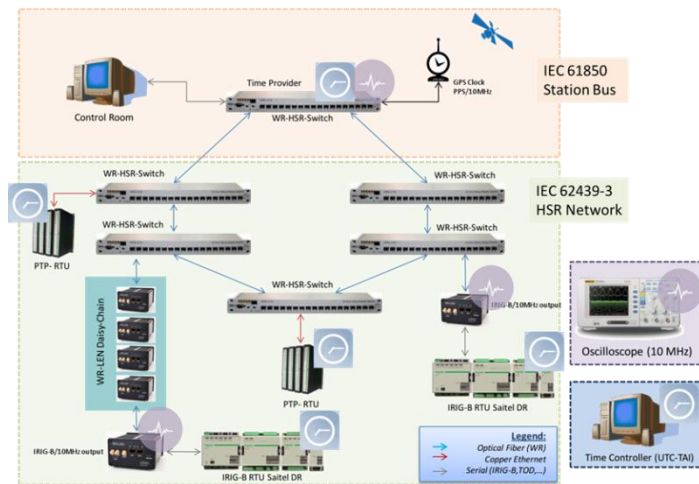


Figure 19: Design of a heterogeneous synchronized low-latency deterministic redundant network using WR-PTP, PTP and IRIG-B

This design meets the requirements to validate the following features:

- Deterministic high-accuracy time and frequency transfer: Using WR as the main time distribution technology provides a precision under 1ns. In addition, WR devices are able to provide time frames in a deterministic way, whilst data frames are disseminated using a best effort approach.
- High-availability of network services: The implementation of redundancy protocols such as HSR in ring networks increases the availability of any of the services in a WR network including time, which are considered critical in Smart Grid.
- Fault tolerance and robustness: HSR increases fault tolerance and robustness since it can avoid single point of failure in the network and, at the same time by sending every frame duplicated, we ensure the reception of critical packets at the destination node.
- Heterogeneous synchronization mechanisms support: Network devices support the distribution of time using different mechanisms for those devices that are not WR-PTP/PTP compatible. WR devices support WR-PTP, PTP, NMEA and IRIG-B.
- Scalability: WR-Switches offer the possibility of connecting thousands of nodes, and daisy-chain configurations using both WR-ZEN and WR-LEN allow maintaining WR-PTP synchronization very precisely in configurations up to 14 hops (~250ps of jitter).
- Safety and security: in addition, safety and security aspects of the ICS have been analysed using the new methodology and tool developed in WP6 by Telvent (Schneider Electric).

For practical reasons, the demonstrator is limited to the number of elements, but it can be significantly increased to illustrate the scalability of the proposed solution.

Industrial Control System

For the final demonstrator, Telvent (Schneider Electric) has implemented an Industrial Control System (ICS) that includes Saitel DR and Saitel DP RTU devices. Based on the typical architecture of an electrical substation, it has been defined a master-slave configuration, where a

set of digital signals are sent from the field site to the front-end level. This architecture and configuration provide us a realistic scenario of validation.

The acquisition system is composed by control devices (SM_CPU866e, HU_A) and I/O acquisition devices (SM_DO32T, SM_DI32, AB_DI, AB_DO). On the other hand, the front-end level is composed by control devices (SM_CPU866e, HU_A) that implement a logic program to send the commands to the acquisition system. In addition, a set of acquisition and supervision signals has been configured for the demonstrator. The acquisition signals are composed by two signals for each protocol. Control protocols such as: IEC 104, DNP or Modbus have been configured for the communication. Respect to the supervision signals, they give us information about the status of the protocols and the configuration.

Concerning the synchronization of the RTU devices, the final demonstrator implements the two most important sources: PTP (for Saitel DP) and IRIG-B (for Saitel DR). Finally, the tool developed by Telvent (Schneider Electric) in WP6 has been integrated in the final demonstrator in order to evaluate the safety and security level, based on the standards IEC 61508 and IEEE 1686.

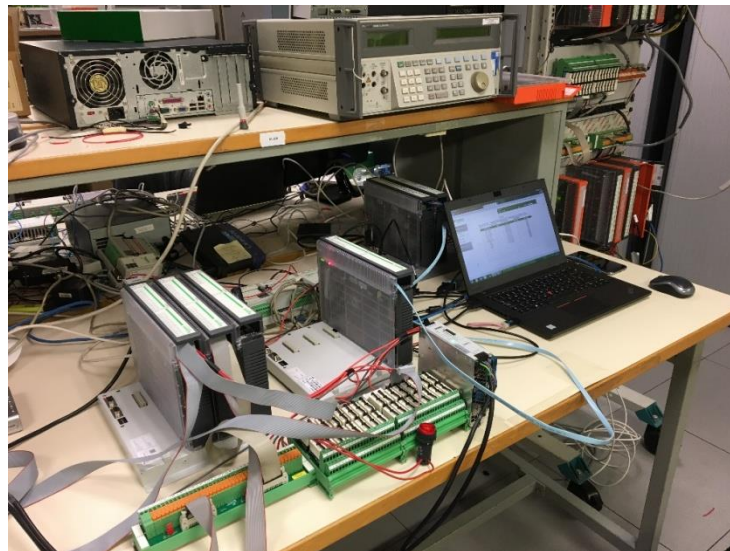


Figure 20: Industrial Control System for final demonstrator

6.3 Results

The final demonstrator configuration (next figure) illustrates the final design previously stated. It is composed by six White Rabbit Switches (WRS), a daisy-chain module with 5 WR-LEN attached, two additional WR-LEN as end-nodes, two PTP RTUs with acquisition modules, two IRIG-B RTUs with acquisition modules and an oscilloscope.

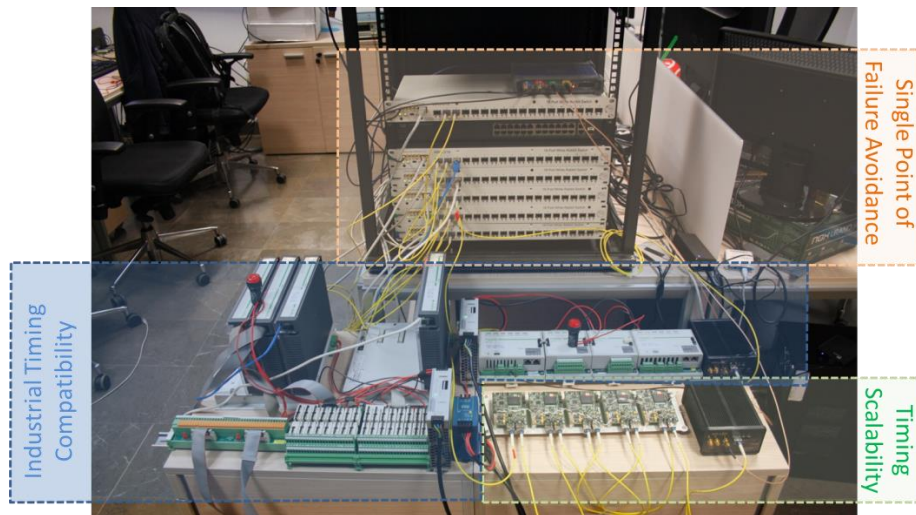


Figure 21: Final demonstrator of the heterogeneous synchronized low-latency deterministic redundant network using WR-PTP, PTP and IRIG-B

The main part of the demonstrator is composed of the six WRS forming a ring network topology, where one of them is the Grand Master timing node that disseminates WR time frames to the entire network. These 6 WRSs form the fault tolerance and redundancy features (HSR) system for both timing and data frames.

The daisy-chain WR-LEN cascade is connected to the ring to also take advantage of the redundancy features and also to enlarge the scalability of the system. The other two WR-LENs are connected, one right after the daisy-chain, and the other one straight to the ring. These two WR-LENs are used as end-nodes to convert the WR timing source into IRIG-B, and send it to the HU_A RTU modules so that they get synchronized to the main timing reference of the network. The other two control RTUs (SM_CPU866e) are directly attached to the ring, where they synchronize using the standard PTP.

All RTUs are connected to the ring, ones using the daisy-chain path, and the others through a WR-LEN node. They are continuously exchanging UDP control frames and even if there is a node failure in the network, control frames reaches their destination thanks to the HSR protocol developed.

Next sections focus on the main results of the demonstrator.

Synchronization Accuracy & Low-latency

White Rabbit is the main technology used to distributed time in the system. It is capable of disseminating time with an accuracy below 1 nano-second, and a precision of 250ps.

In terms of low-latency, by default the WRSs are capable of routing frames with an average of 3us but the utilization of forwarding modules in the WRS for the HSR implementation reduces to the half this latency (~1.5us).

Scalability

SevenS has been working on enhancing timing devices clocks to make the timing system capable of being scalable up to 14 nodes maintaining the WR sub-nano synchronization as next figure states.

Measurement Results

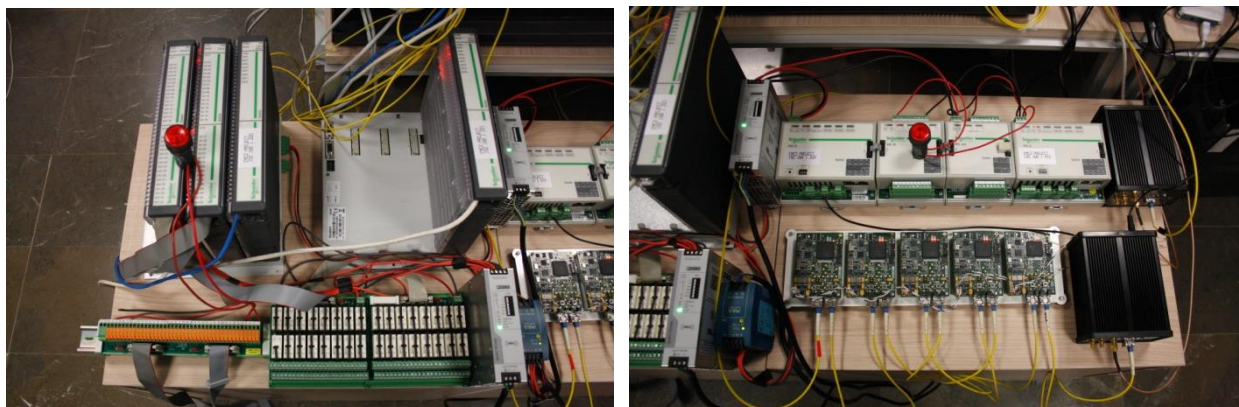
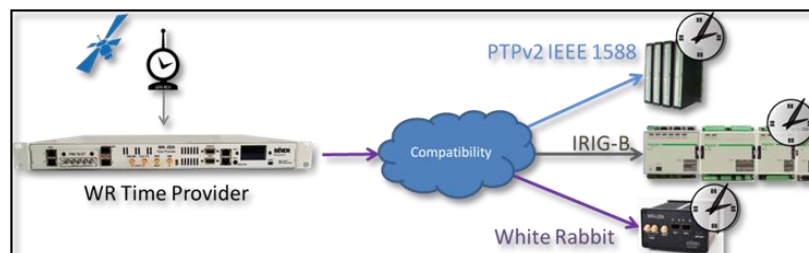
Description	Mean	Std Dev	Max	Min	p-p	Population	Max-cc	Min-cc
TIE1, Ch2, Ch1	-81,951ps	27,014ps	68,750ps	-181,25ps	250,00ps	184705	80,357ps	-80,952ps
Current Acquisition	-59,467ps	12,445ps	-16,250ps	-95,536ps	79,286ps	624	50,000ps	-55,357ps
TIE2, Ch3, Ch1	243,43ps	54,650ps	438,39ps	68,750ps	369,64ps	184795	80,952ps	-79,464ps
Current Acquisition	339,87ps	12,516ps	373,75ps	297,92ps	75,833ps	624	61,250ps	-56,250ps
TIE3, Ch4, Ch1	252,19ps	98,793ps	585,71ps	-37,500ps	623,21ps	184795	78,571ps	-85,417ps
Current Acquisition	194,66ps	13,277ps	240,00ps	157,14ps	82,857ps	624	60,238ps	-60,238ps

Figure 22: WR-LEN daisy-chain synchronization results

TIE1 represents the first node of the chain, TIE2 is the 7th node of the cascade, and TIE3 is the 13th node of the chain. All of them maintain their timing difference below 1 nano-second.

Industrial Timing Compatibility

The aim of the synchronization is to achieve that the RTUs of the demonstrator have the same time reference. Thus, the events of the various subsystems are set with a timestamp as accurate as possible. The synchronization is performed not only at the level of CPU's, but it also becomes involved the acquisition boards containing digital inputs (SM_DI32 and AB_DI). In these modules, changes of the digital inputs can be recorded with a timestamp. In this case, the ICS uses two different sources of synchronization: PTP and IRIG-B. The synchronization source is provided by the Seven Solution equipment. In that sense, several configuration changes and integration tests have been done to check the correct synchronization between devices.

**Figure 23: Synchronization sources. PTP-enabled devices (left), IRIG-B-enabled devices (right)****Single Point of Failure Avoidance: Redundancy**

SevenS has developed and integrated the HSR protocol to provide redundancy and fault tolerance for both timing and data frames using ring topologies. The devices that provide these features are WRS as next figure shows.

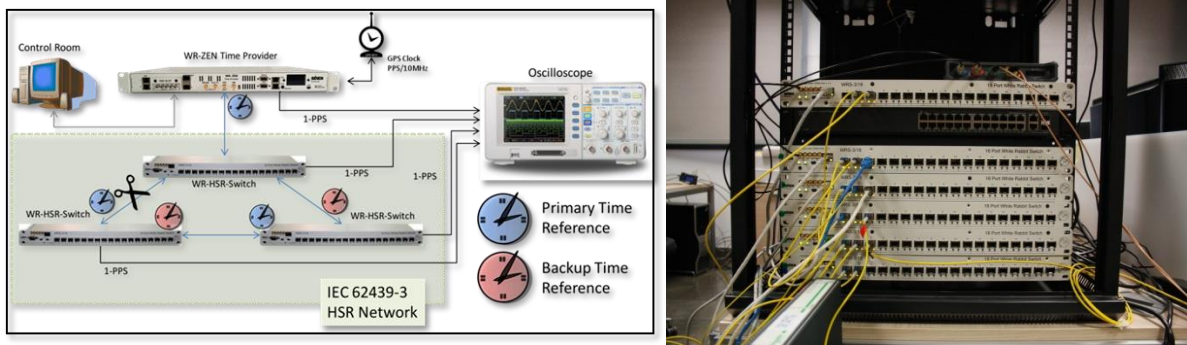


Figure 24: HSR protocol design (left) and implementation (right)

Thanks to this development, we are able to have two different time references, both of them from the same device, but received in two different ports. Thus, we have a primary time reference and a backup one. In case one path fails, the WRSs are able to recover the time reference from the backup source in zero-time recovery. This mechanism is called switchover.

Moreover, normal data is also duplicated and sent into the two paths of the ring, so that any path fails, the data is still received from the redundant path.

Safety & Security

In addition to the synchronization, safety and security aspects of the demonstrator have been considered. Using the new methodology and tool developed in WP6, the Safety Integrity Level estimation of some hardware components and security assessments have been carried out according to the standards IEC 61508 and IEEE 1686.

The WP6 tool includes tables with the Failure Modes of each component analyzed and additional parameters (such as SFF -Safe Failure Fraction-, DC -Diagnostic Coverage- or FITS -Failure In time- among others) that are required to calculate the SIL (Safety Integrity Level) of the hardware. Respect to cyber-security, the tool provides statistics and graphs about the suitability of the system with the security standard. As a result, the tool offers us a safety and security evaluation of the devices respect to the standards.

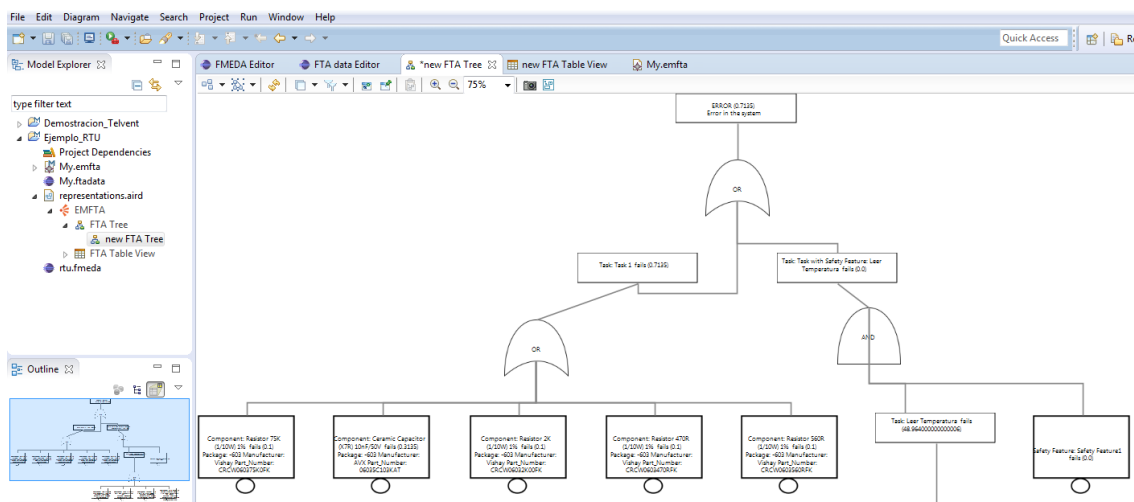


Figure 25: WP6 tool – RTU Fault Tree Analysis (FTA)

6.4 Conclusions

The demonstrator implemented by SevenS and Telvent (Schneider Electric) for this use case represents a new horizon in Smart Grid in terms of timing, single point of failure avoidance, scalability and timing compatibility. Precision time synchronization plays a main role in the Smart Grid. Event analysis, synchrophasor applications or protection system operation require high accuracy time. Thanks to the activities done in this work package, we have improved the synchronization of the Smart Grid using PTP protocol in the RTU and its integration with WR equipment.

Moreover, thanks to the integration of WR as the main time technology, we have been able to synchronize devices of the system with an accuracy below 1 nano-second. By this, we are capable of improving the way Smart Grid acquire and process data, since the precision of event capture and data processing remains in the nano-second domain.

The implementation of redundancy protocols assures single point of failure avoidance and high-availability of both timing and control subsystems that, together with the scalability features, ease the integration of all these described technologies into real Smart Grid systems that demand fault tolerance and large infrastructures support.

On the other hand, the integration of the WP6 tool in the final demonstrator has provided us a safety and security evaluation of the main hardware components. This useful information provides a general vision about the assurance level of the system reducing effort and time for a future evaluation or certification. (More information about this tool and its results is shown in D6.15).

7. REFERENCES

- [1] Bustamante A.L., Molina J.M., Patricio M.A. (2011) Multi-camera Control and Video Transmission Architecture for Distributed Systems. In: Molina J.M., Corredera J.R.C., Pérez M.F.C., Ortega-García J., Barbolla A.M.B. (eds) User-Centric Technologies and Applications. Advances in Intelligent and Soft Computing, vol 94. Springer, Berlin, Heidelberg
- [2] D Marr, T Poggio, "Cooperative Computation of Stereo Disparity", Science New Series, vol. 194, no. 4262, pp. 283-287, 1976.
- [3] Stauffer C, Grimson WEL. Adaptive background mixture models for real-time tracking. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99); June 1999; pp. 246–252.
- [4] Luc Vincent, "Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms", IEEE Transactions on Image Processing, vol.2, pp. 176-201, 1993
- [5] Khan, Saad M., and Mubarak Shah. "Tracking multiple occluding people by localizing on multiple scene planes." *IEEE transactions on pattern analysis and machine intelligence* 31.3 (2009): 505-519.

8. APPENDIX

8.1 Abbreviations

Abbreviation	Meaning
ADMU	Autonomic Decision Making Unit
BE	Best Effort
BTLE	Bluetooth Low Energy
CSAC	Chip Scale Atomic Clock
DUT	Device Under Test
EMC ²	Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments
FOM	Figure Of Merit
HSR	High-availability Seamless Redundancy
IoT	Internet of Things
OSLC	Open Services for Lifecycle Collaboration
P2P	Peer to Peer
RC	Rate Constrained
RT	Real Time
SOC	System On Silicon
SoS	System-of-systems
WAC	WebRTC Application Controller
WLAN	Wireless Local Area Network
WR	White Rabbit
WR-LEN	White Rabbit Light Embedded Node
WRPC	White Rabbit PTP Core
WSN	Wireless sensor networks

9. APPENDIX

9.1 Chromium architecture

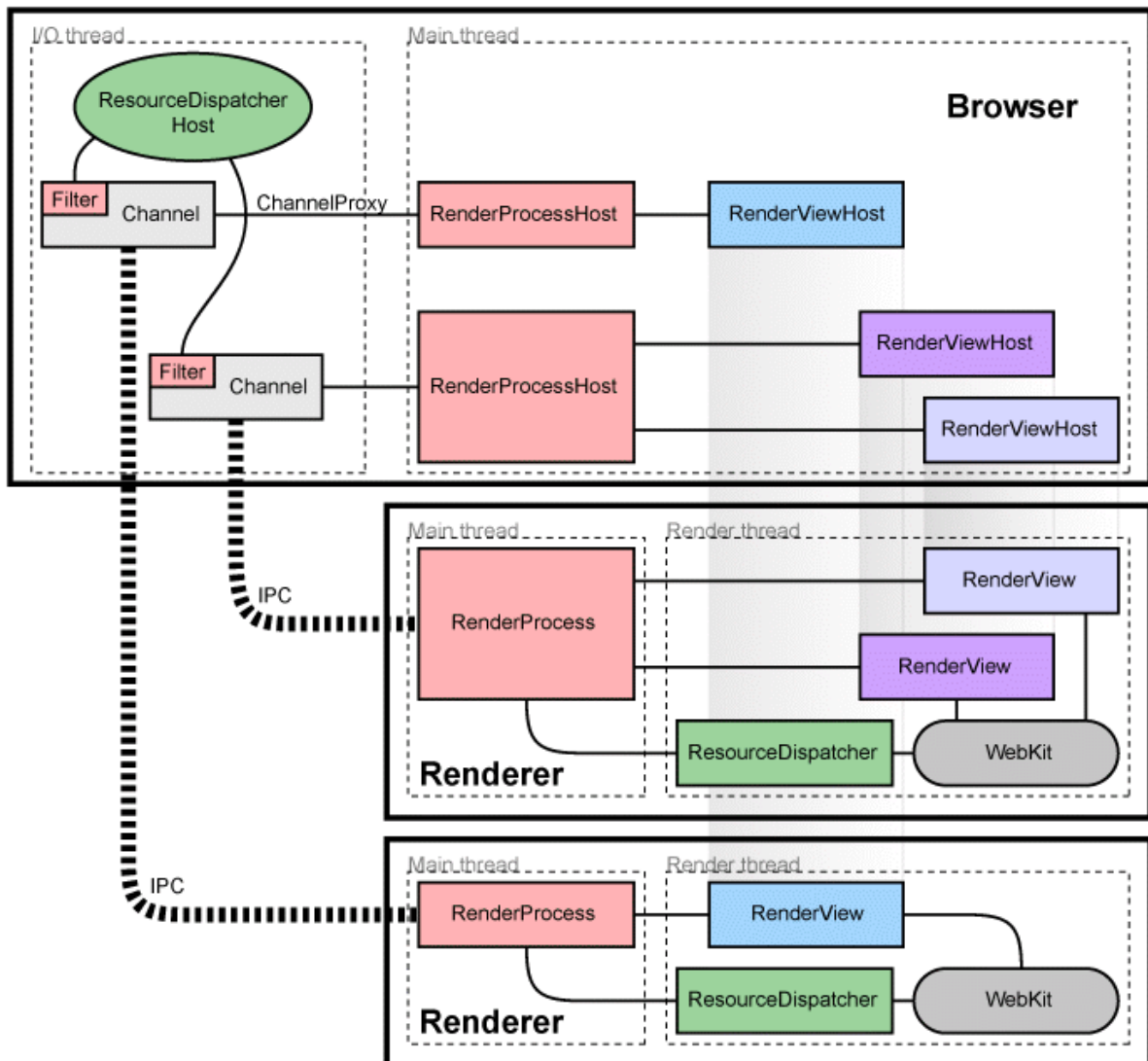


Figure 26: Multi process architecture.

Each render processes has an object **RenderProcess** which communicate with the parent process of the browser. The parent browser has a **RenderProcessHost** per render process to communicate with the render processes. All is communicate with **IPC**

Components and interfaces

In the render process:

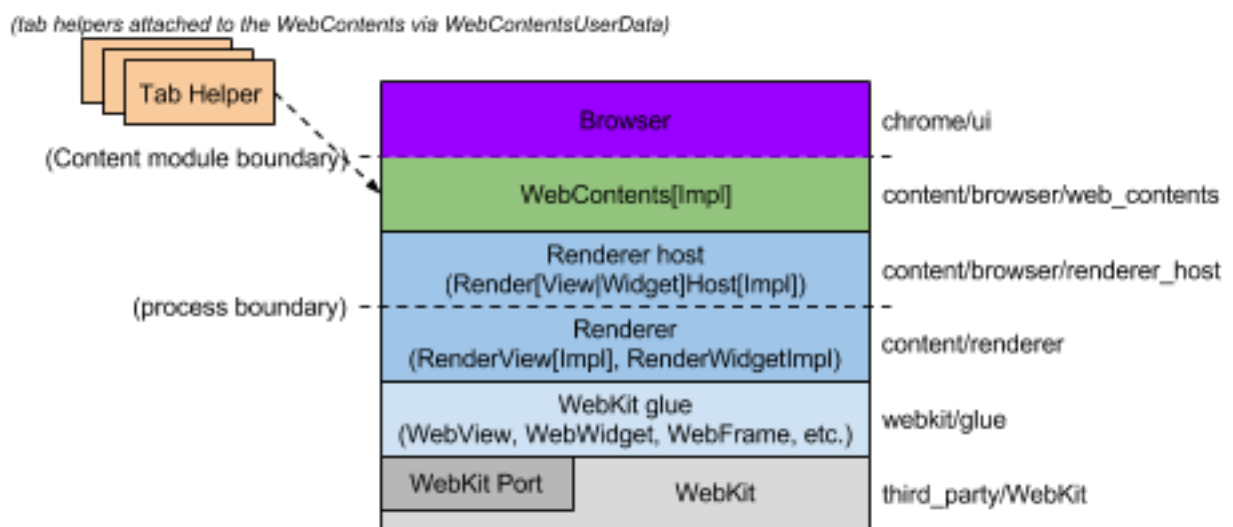
- The **RenderProcess** handles **IPC** with the corresponding **RenderProcessHost** in the browser. There is exactly one **RenderProcess** object per render process. This is how all browser ↔ renderer communication happens.

- The `RenderView` object communicates with its corresponding `RenderViewHost` in the browser process (via the `RenderProcess`), and our WebKit embedding layer. This object represents the contents of one web page in a tab or popup window

In the browser process:

- The `Browser` object represents a top-level browser window.
- The `RenderProcessHost` object represents the browser side of a single browser ↔ renderer IPC connection. There is one `RenderProcessHost` in the browser process for each render process.
- The `RenderViewHost` object encapsulates communication with the remote `RenderView`, and `RenderWidgetHost` handles the input and painting for `RenderWidget` in the browser.

9.2 How Chromium display a web page.



- **WebKit:** Rendering engine shared between Safari, Chromium, and all other WebKit-based browsers. The **Port** is a part of WebKit that integrates with platform dependent system services such as resource loading and graphics.
- **Glue:** Converts WebKit types to Chromium types. This is our "WebKit embedding layer." It is the basis of two browsers, Chromium, and `test_shell` (which allows us to test WebKit).
- **Renderer / Render host:** This is Chromium's "multi-process embedding layer." It proxies notifications and commands across the process boundary.
- **WebContents:** A reusable component that is the main class of the Content module. It's easily embeddable to allow multi process rendering of HTML into a view.
- **Browser:** Represents the browser window, it contains multiple WebContents.
- **Tab Helpers:** Individual objects that can be attached to a WebContents (via the `WebContentsUserData`). The Browser attaches an assortment of them to the WebContentses that it holds (one for favicons, one for infobars, etc).

9.3 Example subWorkers

Example about how main worker creates subworkers, and they receive two numbers in tow event, perform the computation for the range of numbers thus specified, and then report the result back to the parent.

The main page is as follows, it just reports the result:

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Worker example: Multicore computation</title>
</head>
<body>
  <p>Result: <output id="result"></output></p>
  <script>
    var worker = new Worker('worker.js');
    worker.onmessage = function (event) {
      document.getElementById('result').textContent = event.data;
    };
  </script>
</body>
</html>
```

The worker itself is as follows:

```
// settings
var num_workers = 10;
var items_per_worker = 1000000;

// start the workers
var result = 0;
var pending_workers = num_workers;
for (var i = 0; i < num_workers; i += 1) {
  var worker = new Worker('core.js'); // creamos 10 workers diferentes
  worker.postMessage(i * items_per_worker); // les damos el inicio y el fin del "calculo"
  worker.postMessage((i+1) * items_per_worker);
  worker.onmessage = storeResult;
}

// handle the results
function storeResult(event) {
  result += 1*event.data;
  pending_workers -= 1;
  if (pending_workers <= 0)
    postMessage(result); // finished!
}
```

The subworkers are implemented as follows:

```
var start;
onmessage = getStart;
function getStart(event) {
  start = 1*event.data;
```

```

    onmessage = getEnd;
  }

  var end;
  function getEnd(event) {
    end = 1*event.data;
    onmessage = null;
    work();
  }

  function work() {
    var result = 0;
    for (var i = start; i < end; i += 1) {
      // perform some complex calculation here
      result += 1;
    }
    postMessage(result);
    close();
  }

```

9.4 Other types of workers

Shared workers

Shared workers can be shared worker thread between any window of the same origin/domain. The shared workers are too similar in code to Workers. The main difference is that it is necessary to use *onMessage* to process messages in the object port of shared workers, for example.

```

var sharedWorker = new SharedWorker("SharedWorker.js");

  sharedWorker.port.onmessage = function (event){
  ...
}

```

Also, it's possible to use *addEventListener*, but in this case is necessary to use *start()* method on port object. For example:

- In main page:
`sharedWorker.port.postMessage("Message to worker");`
- In shared worker:
`sharedWorker.port.addEventListener("message", onMessage, false);`
`sharedWorker.port.start();`

Service workers

Supported by opera, mozilla and chrome. HTTPS it's necessary to use them. Service workers use the API of workers, but permits to install in the browser, creating a kind of application. The application can be accessible even without connexion, and can be work as Cache, desktop notification, etc.

The following links show the Workers and Web Workers active in chrome browser.

<chrome://inspect/#workers>

<chrome://inspect/#service-workers>

In general browsers are enabled Service Workers, but some characteristics can't work. If this happened, it is necessary to activate the experimental API. The experimental API can be activated:

- Firefox Nightly: Go to `about:config` and set `dom.serviceWorkers.enabled` to `true`; restart browser.
- Chrome Canary: Go to `chrome://flags` and turn on `experimental-web-platform-features`; restart browser (note that some features are now enabled by default in Chrome.)
- Opera: Go to `opera://flags` and enable `Support for ServiceWorker`; restart browser.

9.5 ServiceWorkers lifecycle.

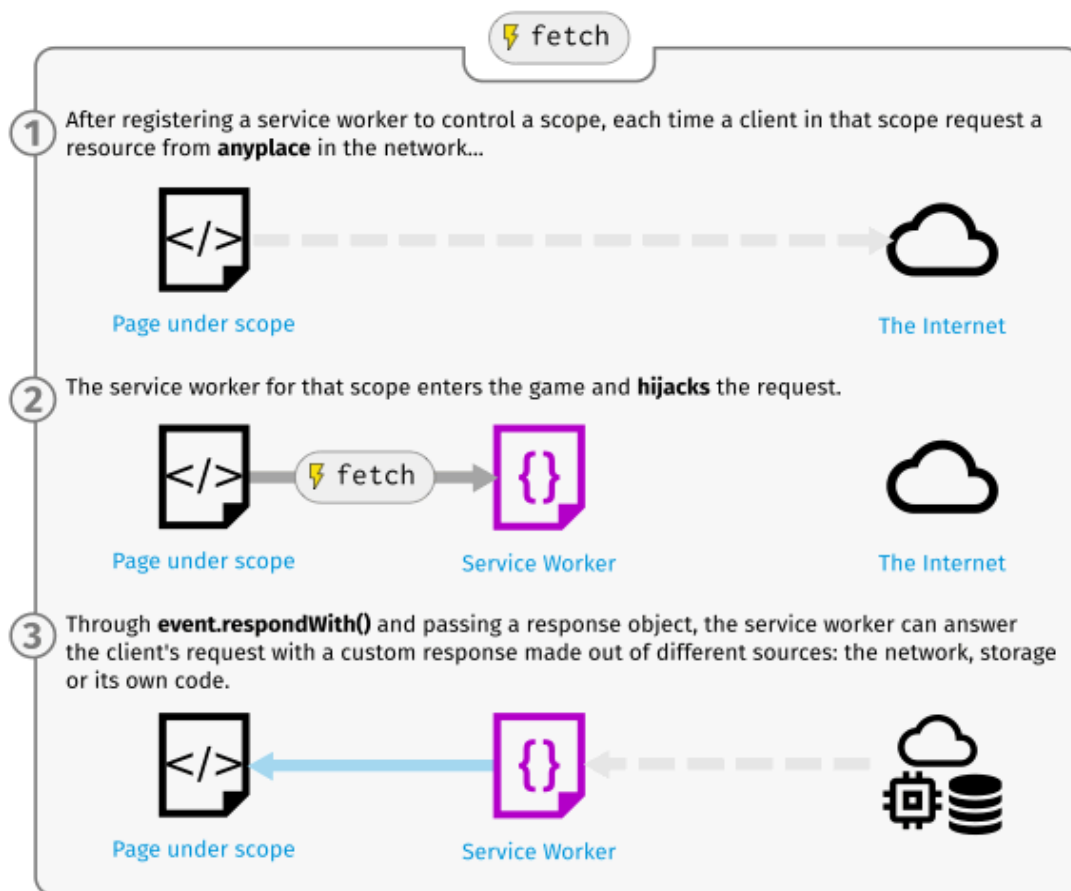


Figure 27: Life cycle of a ServiceWorker.

9.6 Service Worker use case.

The most common use of a Service Worker is like a cache, because Service Workers have access to the database, and have information in case of lost connection. The behaviour allow to recover information while you are offline. Service Workers allow higher control about all action than

caches. The behaviour of Service Workers is like a man in the middle, staying between the page and the server. As the following picture shows.







The Service worker capture all HTTPS request, allowing to control it, and control the response. The response can be from the cache, forwarding the request to the destination server and updating the cache with the new result. It's possible to modify the HTTPS header adding or deleting elements, etc. Service worker permit all control options that permits anyone control like a HTTPS server.

The most common errors in Service workers are:

- Navigation over HTTP and not over HTTPS.
- Invalid scope, the root scope will be the directory where stay the Service Worker, except if you include another scope in the constructor.

9.7 Web Components with workers

Web components that use workers:

 platinum-bluetooth	A set of Polymer elements to discover and communicate with nearby Bluetooth devices	★ 15	🔗 4
 platinum-https-redirect	Force a redirect to HTTPS when not on a local web server.	★ 3	🔗 4
 platinum-push-messaging	Push Notifications with Polymer	★ 16	🔗 7
 platinum-sw	Service Worker Polymer element	★ 41	🔗 11

Platinum-Bluetooth and platinum-push-messaging use Service Workers to offer push notification and communication with Bluetooth devices even without connexion. Platinum-sw is a web component that allows the abstraction of much complexity of the Service worker.

9.8 Testing

Raw computing capacity

Objective

The goal of this test is to get insight about how concurrency affects to raw operations with high computational load. The experiment performs a mathematical operation that consist of finding narcissist numbers in a interval.

Procedure

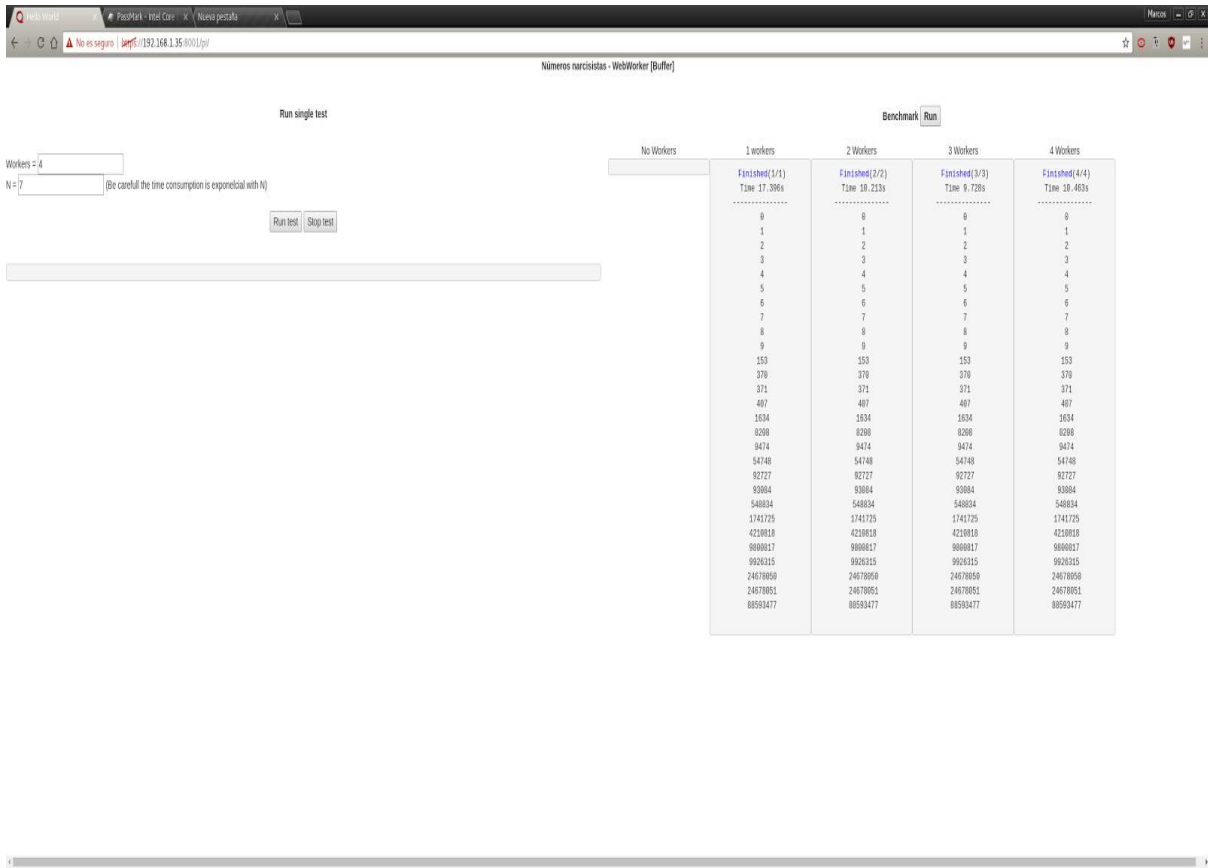
The experiment runs a single test with a customizable interval and workers or a benchmark which runs tests in a row with different number of workers but keeping the interval.

To allow concurrency, the interval is spitted in as many parts as workers, and each worker operates in one interval.

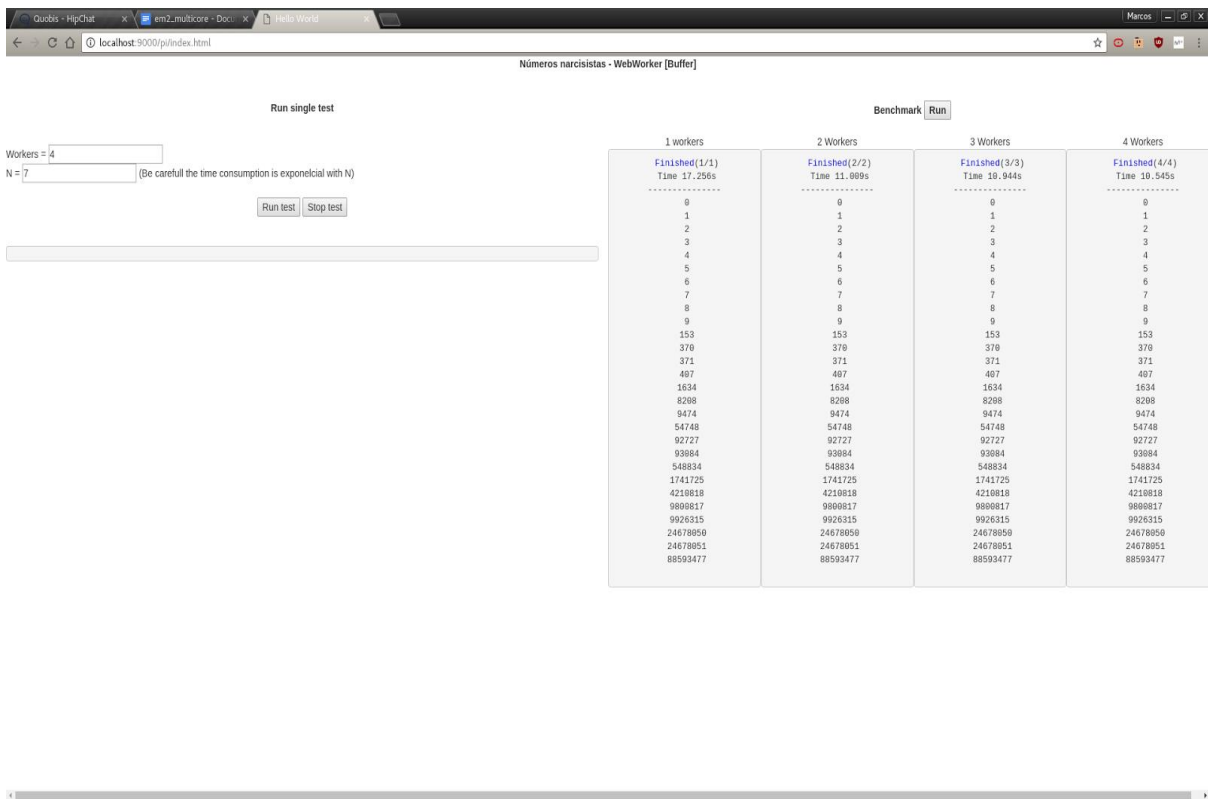
Results - Desktop

Environment:

- Chrome 56
- Linux
- Intel Core i5 4200H (2 physical cores, and 2 logical cores per physical)



Results 1



Results 2

The results shows how the performance increases with the number of workers. The difference between one or two workers are considerable, but more than this, the increase is less, because the laptop where the benchmark is passed have four logical cores, that implies that the two last cores are shared between the workers and the rest of the process of the operating system. This is the reason because the performance with four workers are in some cases worst than three.



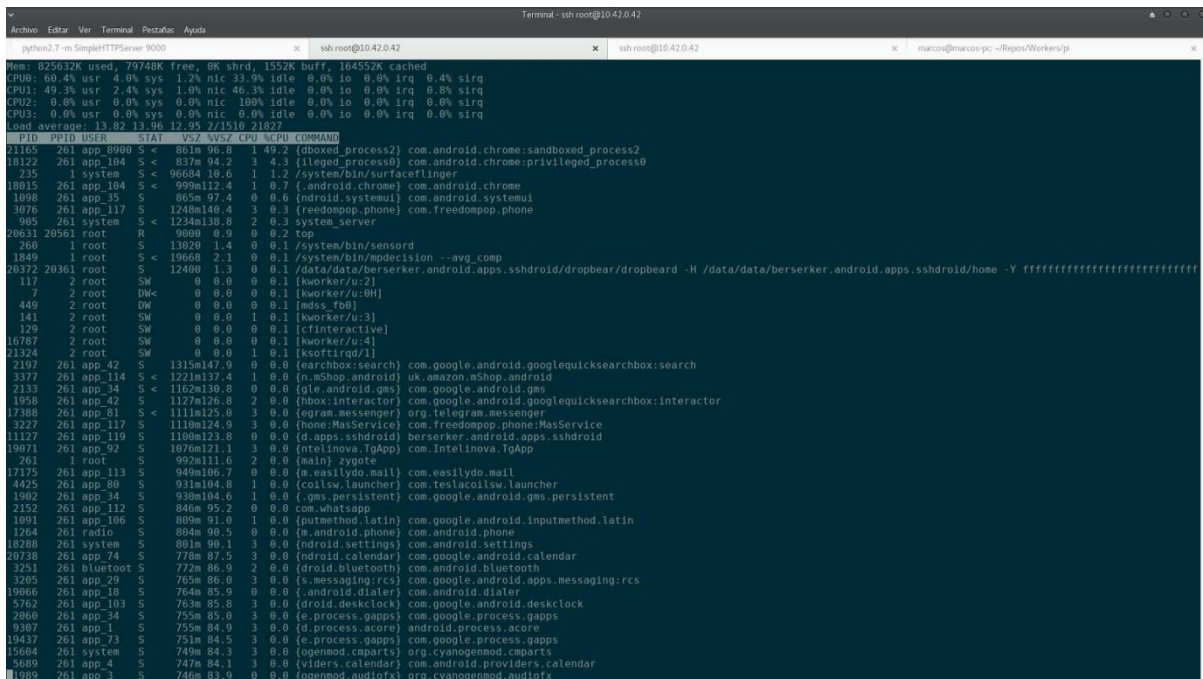
Output of “top -H”

The behaviour previously commented can be checked in the capture, the two dedicated worker use 95/96% of the CPU, but the last two ones use 92% of the CPU because the cores are shared with the SO.

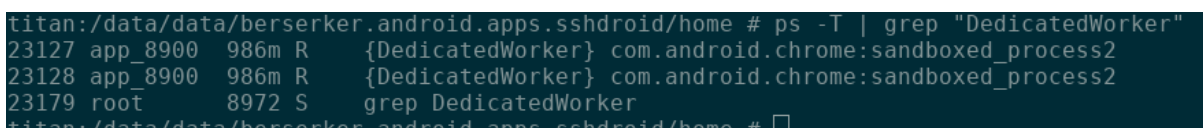
Results - Mobile

Environment:

- Chrome 56
- Android 7.1.1
- Motorolo Moto G 2014

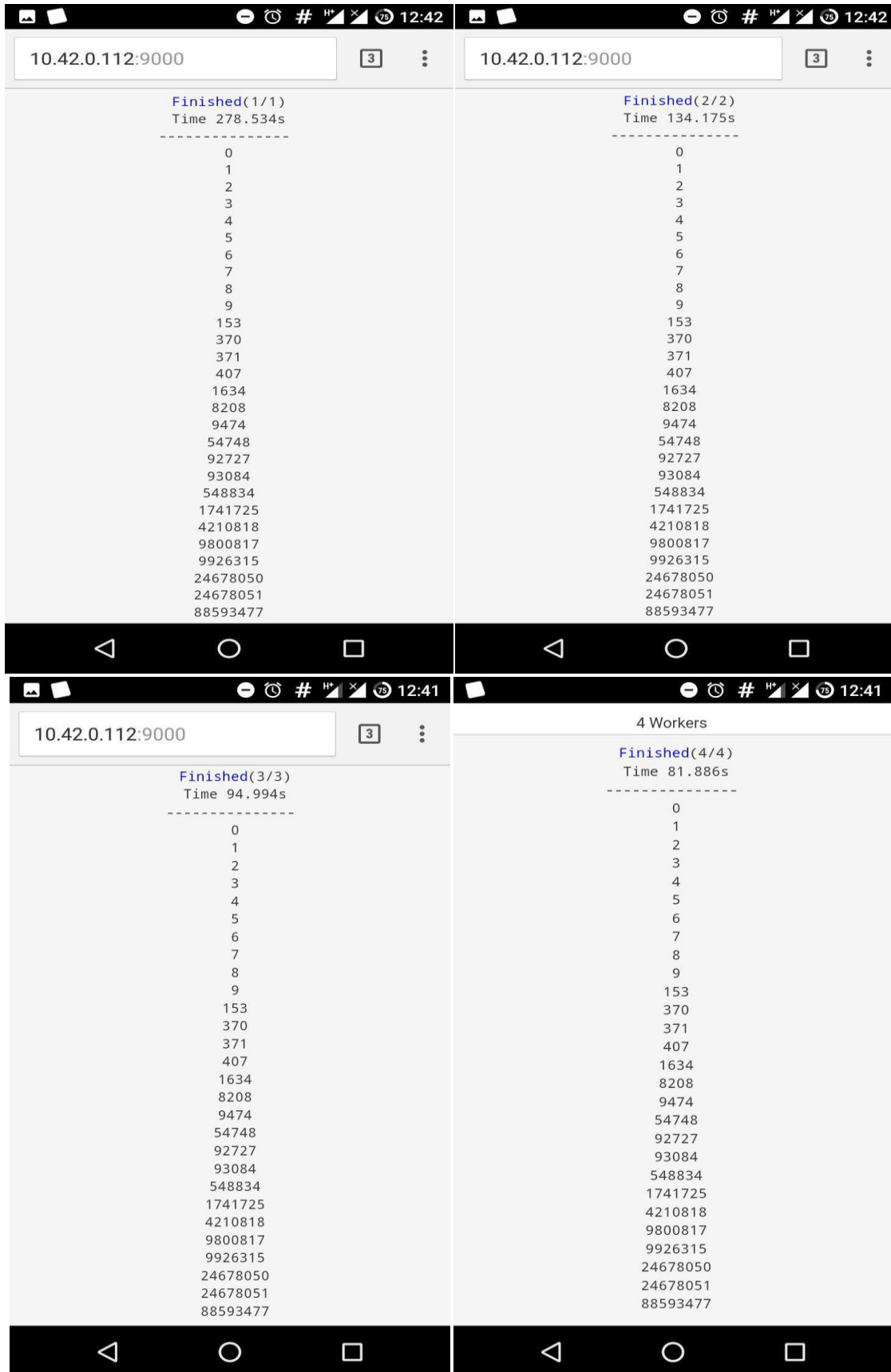


Output “top” command in mobile.



Output “ps -T | grep “DedicatedWorker” command

As the first pictures shows, dedicated workers can't be seen with top like in desktop because "top -H" isn't available on Android. But if "ps -T | grep "DedicatedWorker"" is executed in terminal, we can see the worker threads, like the second picture shows.



The results shows the huge increase of performance with workers. With one worker the benchmark needs around 300 seconds, whereas with two ones the needed time is 150 seconds and 100 seconds with three workers. This increment is almost linear with one, two and three. This not happen with four, because the smartphone has four cores, then the last one is shared with rest of SO and apps in the mobile. This is clearly visible when you try to do scroll, and the web page is freezing, in some cases the freezing becomes in a close of the webpage caused by the high use of CPU.

File transfer

Objective

The goal of this experiment is check the viability of workers in a common task. In this case the task is a file transfer.

How To

The test compares two approaches:

- to perform the transfer in the main page
- to perform the transfer in workers

Both of them show the time needed to complete the task. In both scenarios the main page has a workload of around 100% of use.

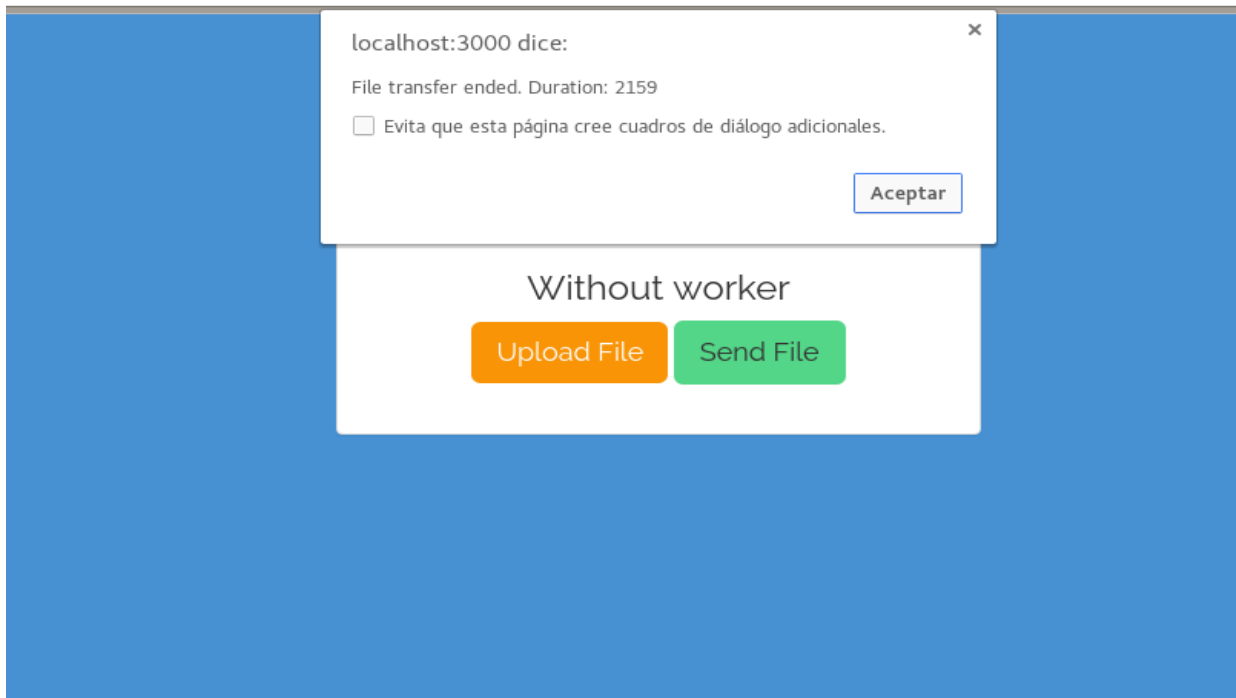
Results

Environment:

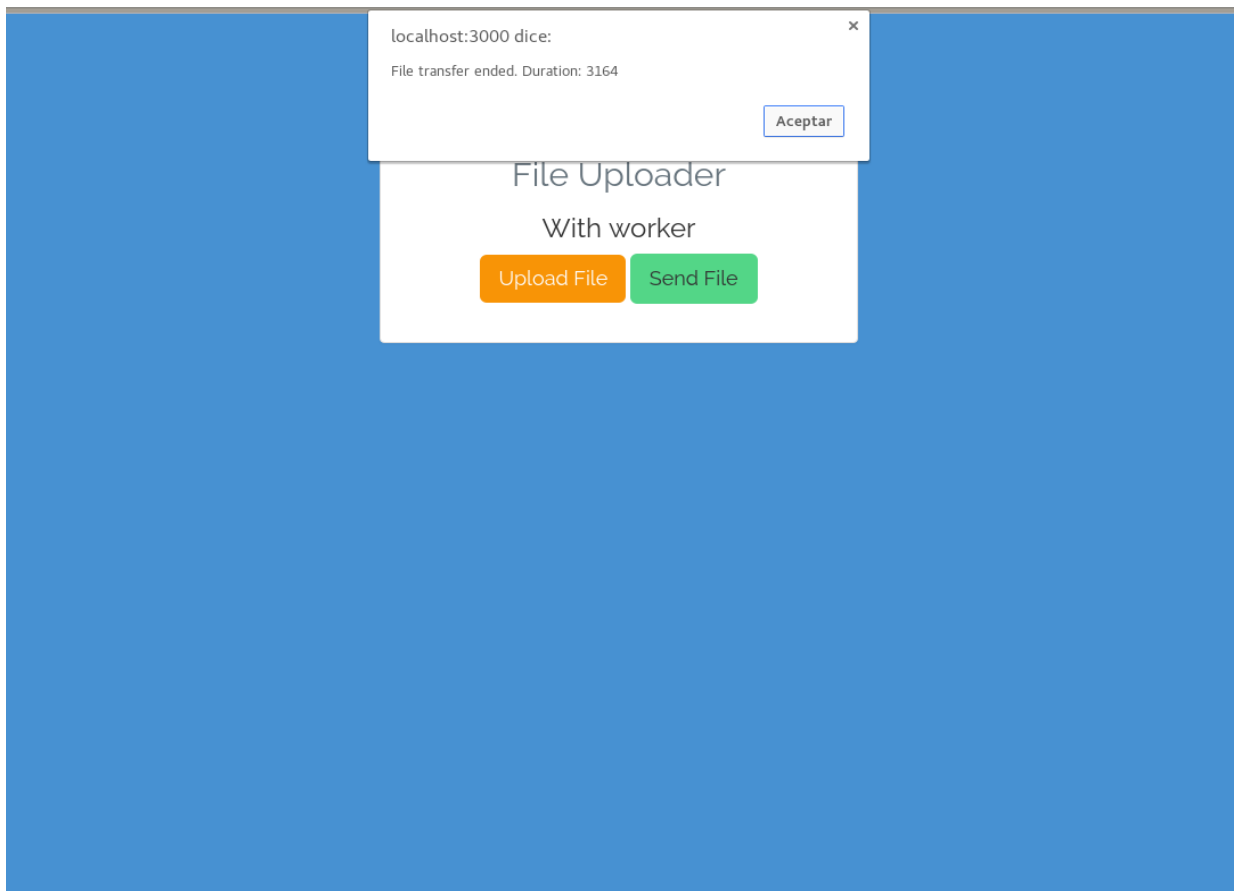
- Chrome 56
- Linux
- Intel Core i5 4200H (2 physical cores, and 2 logical cores per physical).

The next pictures shows the total time to do the transfer around 60MB of photos. The first picture shows the time needed without worker, whereas the second one represent the time with worker.

As the picture shows, more time is needed with worker than without it. This is because Chrome has a built-in mechanism to manage transfers as a separate threads at SO level. However, there is still a pending question: why does the worker need more time to perform the transfer than the approach without it? This is caused because the worker has to communicate with the main page, and the workload of the main page delays the messages and the events between the two parts.



File transfer without worker.



File transfer with worker.

10. APPENDIX: DISTRIBUTED PASSENGER TRACKING IN DETERMINISTIC NETWORKS

Latest advances in hardware technology and state of the art of computer vision as well as artificial intelligence research can be employed to develop autonomous and distributed monitoring systems [1]. Reliably detecting a crowd of pedestrians is challenging, mostly due to the severe occlusions that occur when many people are involved. The occlusions may cause missing detection, wrong estimation of the count of people. However in many cases, it is possible solve the occluded scenario by providing assistance information (denoted as metadata), such as the detections from another camera which is observing the same scene from a different view. In camera networks the exchange of metadata may not arrive on time because of limited network bandwidth and the non-deterministic behaviour of Ethernet. The rate-constrained (RC) TTEthernet provides the requested quality of service by a deterministic behaviour and short queuing delays. In this way, various visual sensors watching the same scene can exchange information and support each other to solve the passenger tracking.

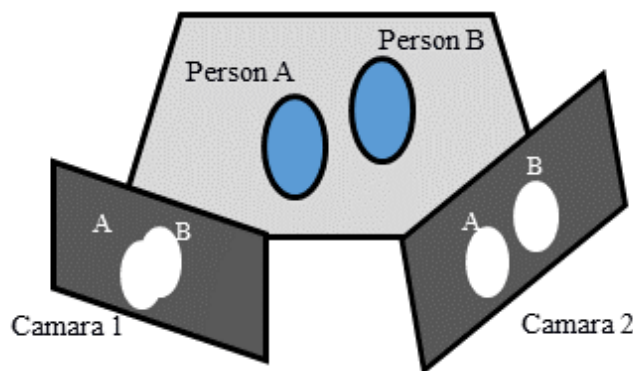


Figure 28: People A and B are occluded in the view of camera 1, while not occluded in camera 2.

As Marr [2] stated about human visual perception: If two objects are separated in depth from the viewer, the relative positions of their images will differ in the two eyes. Our brains are capable of measuring this disparity. This concept applies to the multiple camera recordings, too. As Figure 28 shows, the relation between projected locations of the persons can vary by different camera settings, which means multiple cameras are able to complement each other by communicating their detection results (metadata). By using several cameras with deterministic network communication, we can improve the detections for the occlusion scenarios. To demonstrate the advantage of open deterministic networks in vision surveillance application, we implement a framework of detecting pedestrians using multiple cameras, which are equipped with RC communication channel. In this demonstration, the local processing unit receives the detections from the sender processing unit, therefore resolves the occlusions and improves the detection results.

10.1 Description of final demonstrator

As Figure 29 shows, each local sever gets a surveillance video stream from its connected camera. The individual processing unit performs people detection algorithm which results in a collection of bounding boxes and its total count (metadata). As the multi camera set-up was calibrated beforehand, the images of the different cameras are related by a homography, which can be used to register the detected bounding boxes from multiple cameras in one scene. The metadata contains the size and coordinates of each detected bounding boxes and are transmitted via a RC communication channel.

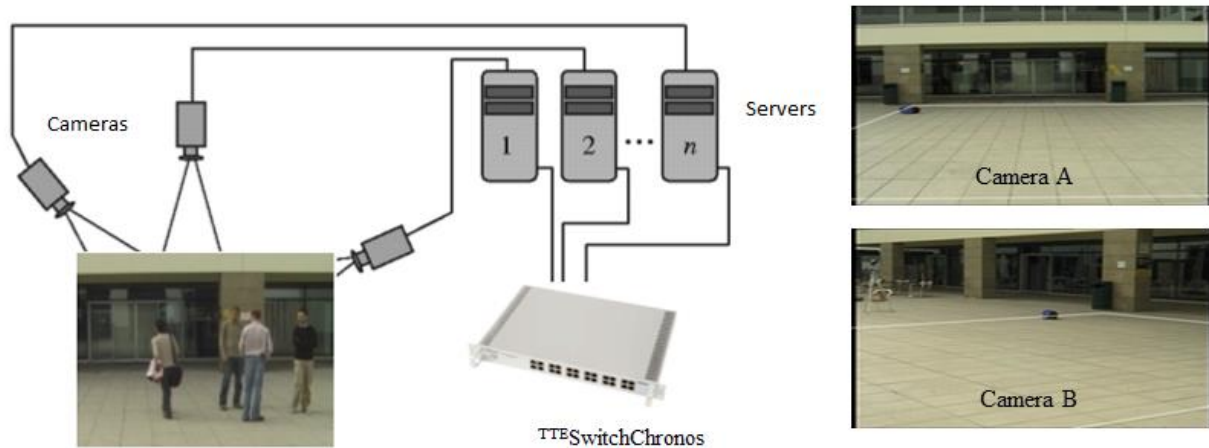


Figure 29: Setup of Multi-Camera Passenger Detection Using Open Deterministic Network.

As exemplarily displayed in Figure 29, the test data comprises of video recordings from multiple (camera A and camera B) cameras, recorded in an open space in front of an entrance door. The white line on the screenshots shows the limits of the area that we defined to obtain our detection results. The blue spots show the location of the cameras. In this set-up, camera A points to the front view of the entrance door, while camera B looks to the sideways view of the entrance.

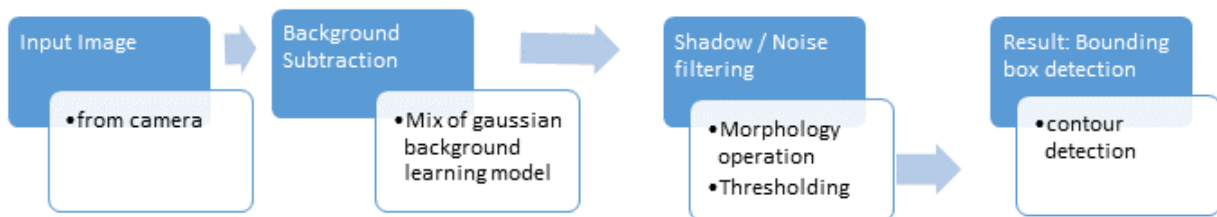


Figure 30: Overview of the proposed passenger tracking.

Each individual camera performs the people tracking in following process (Figure 30 and Figure 31): First it performs background subtraction (foreground detection). It approaches by modelling each pixel as mixture of Gaussians [3] and uses an online approximation to update the model. In this technique, the processing unit reads the frames and estimates the background while learning the current frame. The second step is to generate the region of foreground segments which represent the pedestrians. With morphology operations, we eliminate small noise (holes). With thresholding operation [4], we eliminate the shadow segments. Finally by finding the contours,

the individual pedestrian is fit into a rectangle bounding box, contributing to the detection result [5].

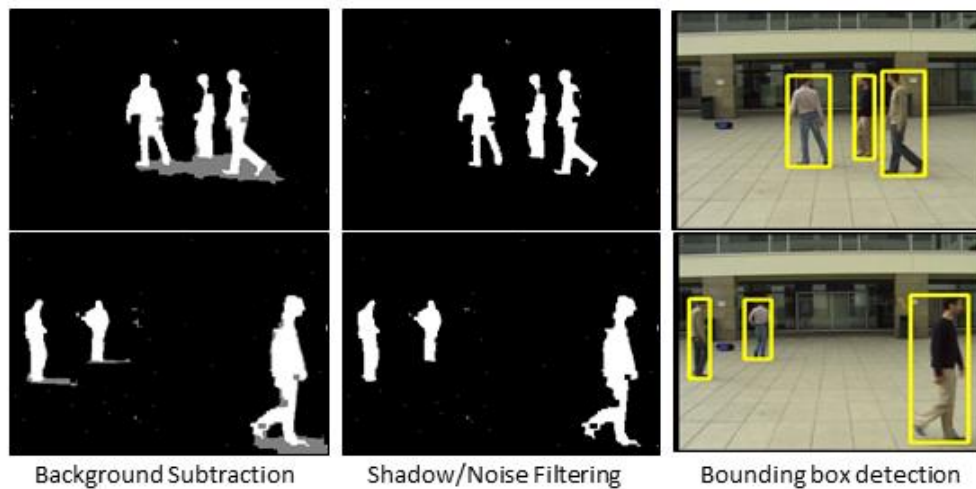


Figure 31: Algorithm flow of people detection for individual camera

10.2 Results

The ideal person detection algorithm is able to detect each person as an individual, which we mark as a rectangle bounding box. For certain scenarios, however, a camera may lose the detections due to occlusions.

In Figure 32 this difficulty is exemplarily presented. Camera B at frame no. 376, detects 2 bounding boxes for a scene with 4 persons, in which one bounding box includes 3 persons. That is because the person wearing a black shirt heavily occludes the persons walking behind him. At the same moment, camera A successfully detects 4 individual persons in the same scene, because at this view point, the persons are quite separate and does not result in a false detection.

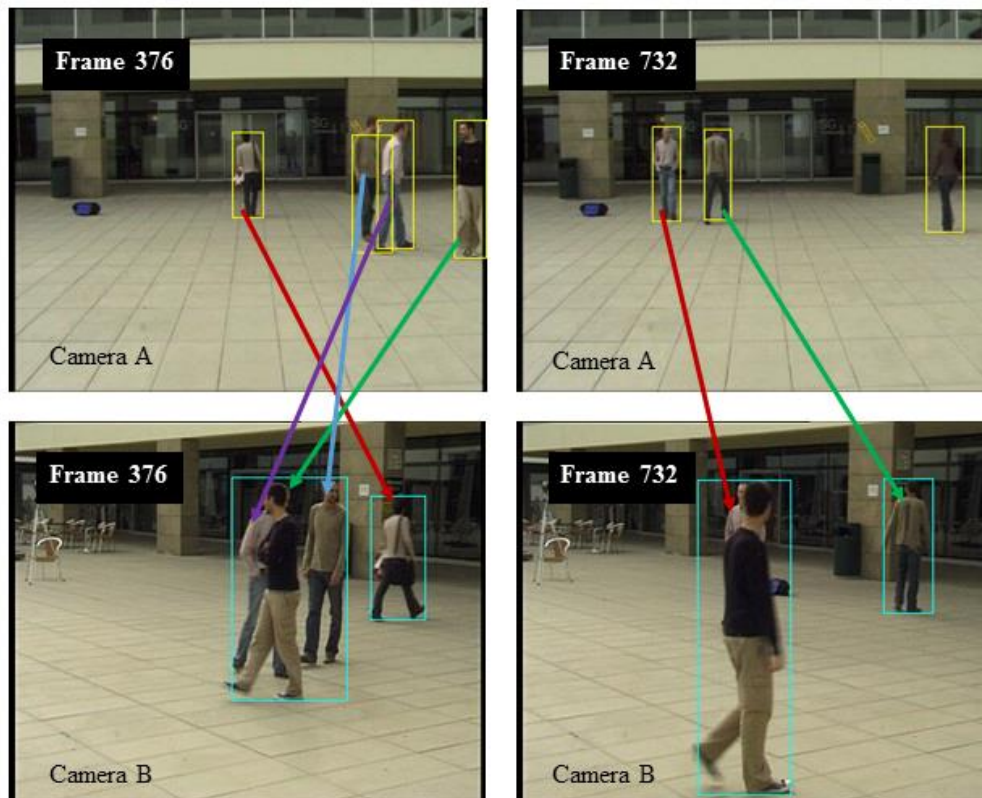


Figure 32: Camera A detects 4 persons in frame 376 and 3 persons in frame 732. This metadata helps camera B, which locally only detects two persons in both frames, to resolve the occlusion of persons

The detection results from camera A are transferred through RC link to camera B in order to resolve the occlusions as depicted in Figure 32. Another advantage is that even information not relevant yet can be used to better predict a person appearing in the scene. For instance, in Figure 32 frame 732, the lady walking on the right side of camera A is not covered by camera B, which means, camera B is impossible to detect her due to limitation of covering area. Since this information is exchanged, camera B can be prepared that this lady may be visible in the near future. Merging the information of all cameras provides a comprehensive overview of the current scene.

10.3 Conclusions

Complex computer vision tasks such as passenger tracking can be only reliably solved by distributed computer vision solutions. In this case, multiple cameras monitoring the same scene are able to perform better if the detection results are exchanged between cameras. The data exchange of the cameras is based on RC links, which ensure a deterministic network behavior. Hence, the quality of the distributed algorithms is not affected anymore by the transmission of data as experienced in ordinary Ethernet networks.

11. APPENDIX: USE CASE 5 DEMONSTRATOR DEVICES

11.1 WR-Switch



White Rabbit Switch is the key component of the White Rabbit Protocol that provides precision timing and high synchronization over an Ethernet based network. WR-Switches implement the redundancy protocol HSR, which development has been covered in WP1, Task 1.6. HSR requires the implementation of Peer-to-Peer (P2P) and PeerDelay mechanisms to propagate the clock and calculate the delay between nodes in ring topologies, which have been part of WP4, Task 4.3. These features made possible to increase high-availability of timing and the avoidance of fault in case of node failure.

11.2 WR-ZEN



The WR-ZEN Time provider is based on the White-Rabbit technology and targets stand-alone and dependable applications. It is able to support daisy chain configurations. Optionally, it can include CSACs (Chip Scale Atomic Clocks) oscillators for applications requiring long time holdover. It has redundant power supplies as well as pluggable hot-swappable fans and system status (heating and power supply) monitoring support.

11.3 WR-LEN



The WR-LEN (White Rabbit Light Embedded Node) is based on the White-Rabbit technology and targets stand-alone applications. WR-LEN is able to support daisy chain configurations based on two SFP ports as well as tree topologies using WR switches. This development together with QoS features have been developed within WP4, T4.3.

11.4 Saitel DR platform



The Saitel DR platform has been used in the UC5 demonstrator for the IRIG-B synchronization. The time received from the IRIG-B device is used to put in hour the clock of the system and the RTC. The formats configured in the demonstrator are: IRIG-B003 and IRIG-B006.

11.5 Saitel DP platform



The Saitel DP platform has been used in the UC5 demonstrator for the PTP synchronization. In this case, the synchronization device works, according to the IEEE-1588 terminology, as an "Ordinary Clock" (OC) in its modality of "Slave-only". An OC can only have associated one physical port or Ethernet interface. By means of the elements PTP_CLOCK and PTP_PORT are configured the different options of the OC and of its associated ports.