International Journal of Intelligent Robotics and Applications (2023) 7:304–320
https://doi.org/10.1007/s41315-023-00281-3

REGULAR PAPER

WebRTC-based MOSR remote control of mobile manipulators

Allal Tiberkak[1] · Abdelfetah Hentout[2] · Abdelkader Belkhir[3]

Received: 11 March 2022 / Accepted: 11 April 2023 / Published online: 25 April 2023

Abstract
This article describes a contribution to the field of telerobotics via the Internet through the development of a web-based platform allowing the remote control of robots by multiple users, simultaneously. It also deals with minimizing the execution times of tasks by reducing connection and interaction delays. For this purpose, the *Web Real-Time Communication* (*WebRTC*) technology is utilized. The developed remote manipulation system allows the operators to visualize the robot, its surroundings and the data incoming from its sensors, and to carry out basic tasks, either independently by the manipulator or by the mobile robot, or jointly by both mechanical sub-systems. In addition, to ensure the control of the remote robot by several operators simultaneously, a priority system managing parallel tasks and a chat system between the operators have been proposed. Besides, many teams are able to exploit the robot, concurrently. The WebRTC-based *Multiple Operator Single Robot* (*MOSR*) telerobotic platform is validated on the emulator of the *RobuTER/ULM* mobile manipulator through various scenarios of primitive tasks over the Internet.

Keywords Telerobotics · Web Real-Time Communication · Multiple Operator Single Robot

1 Introduction

With the increasing use of the Internet, Internet-based telerobotics, which includes controlling distant robots via a web browser, is becoming a very intriguing and promising subject of research worldwide (Sayouti et al. 2011). Internet-based systems allow quick and low-cost relocation of users, and do not rely on the equipment location that controls the remote robot. The development of a web interface enables researchers to work and collaborate to control the robot from any place worldwide (Elkady and Sobh 2009).

One of the first remotely robots is *Mercury Project* (Jara et al. 2011) put online on August 1994 at the University of Southern California (Crainic and Preitl 2014). This evolved in the *Telegarden Project* and the system from the University of Western Australia (Yu and Huang 2011), in which a user is able to manipulate different objects using the remote arm. It consists of a *SCARA* robot fitted with a CCD camera over a semi-annular workspace containing sand and buried artifacts (Santoso et al. 2018). Just after, a 6-Degree-of-Freedom (DoF) manipulator was put online at the same university in September 1994 (Dalton 1998). This allows users to operate a robot placed above a table with wooden blocks positioned on it (Dalton 2001). Another work, *CINEGEN project*, eases simulation aspects and non-specialists offline programming of manipulators (Flückiger et al. 1998). Finally, *PumaPaint Project* (Stein 2003) is the collaboration output between Wisconsin University and Wilkes University (Sayouti and Medromi 2012). It is an online manipulator allowing a user with a Web browser to manipulate a *PUMA760* robot for coloring or painting on white papers using real brushes and paint (Aghili 2013).

Other scholars were interested in Internet-based telerobotics allowing a user to remotely control mobile robots in

✉ Allal Tiberkak
allal.tiberkak@gmail.com

Abdelfetah Hentout
ahentout@cdta.dz

Abdelkader Belkhir
belkhir@lsi-usthb.dz

[1] Department of Mathematics and Computer Science, Faculty of Sciences, University Dr. Yahia Fares of Medea, Boulevard de l'ALN, Ain D'heb, 26000 Medea, Algeria

[2] Division of Robotics and Industrial Automation (DPR), Centre for Development of Advanced Technologies (CDTA), BP 17, 16303 Baba Hassen, Algiers, Algeria

[3] Computer Systems Laboratory, Faculty of Electronics and Computer Science, University of Sciences and Technology Houari Boumediene (USTHB), BP 32, 16111 Bab Ezzouar, Algiers, Algeria

static, dynamic or uncertain environments (Emharraf et al. 2016). *Xavier* is an online autonomous indoor mobile robot at the Carnegie Mellon University (Burgard et al. 1998). This robot receives commands to move to various offices in a building and broadcasts camera images, simultaneously. *WebPioneer project* (Grange et al. 2000) is another web system where the remote operator drives a *Pioneer* mobile robot in dynamic environments. *KhepOnTheWeb* robot (Simmons et al. 2001) allows users to control a *Khepera* mobile robot evolving inside static workspaces (Moutaouakkil et al. 2010).

Some researchers dealt with telerobotics of mobile manipulators via the Internet. Carelli et al. (2008) proposed a teleoperation system for remote control of a three-wheeled mobile robot surmounted by a 5-DoF manipulator. Authors in *RISCbot project* (Elkady and Sobh 2009) developed a Web-based teleoperated robot via sensor fusion. This robot is composed of a wheelchair mobile robot with an end-effector to manipulate objects. Khiter et al. (2012) proposed a multi-agent Internet-based telerobotic platform for mobile manipulators. The validity of this architecture has been demonstrated through several primitive operations accomplished by *RobuTER/ULM* (described in subsection 4.1).

The aforementioned works showed that telerobotic control over the Internet as a communication channel yields more issues such as unavailability, inaccuracy and instability. This is largely caused by the Quality of Service (QoS) on unsecured Internet and notably by unstable delay from end-to-end sites (Witrant et al. 2004; Cloosterman et al. 2009; Mostefa et al. 2015). It is also noticed that a few work only deals with remote robots via the Internet, especially for mobile manipulators. Despite the fact that these robotic systems performed well their tasks, some weaknesses could be identified:

- For most of them, communication rate between users and robots is high. In addition, these systems do not provide/support a good video quality.
- All these online telerobotic systems require specific software installations or tools to be exploitable.
- Most of them are *Single Operator Single Robot* (*SOSR*), where the control is limited to one operator at a time.
- Operators are forced to work sequentially and wait in a queue. This may lead to time loss during tasks execution.
- Online robots must be developed in such a way that non-specialists can run them using simple interfaces and that they are available 24 h a day (Goldberg et al. 2002).

In this context, this article presents a generic, yet appropriate solution that eliminates most of these imperfections. Hence, it describes the development of a Web-based platform to facilitate access and control of remote robots in *Multiple Operator Single Robot* (*MOSR*) mode. The application relies on Web technologies and provides real-time communication between the various entities of the robotic system. This is realized thanks to the exploitation of *Web Real-Time Communication* (*WebRTC*) technology that offers many benefits such as (*i*) providing a real-time communication between the system components, (*ii*) ensuring the streaming of any kind of data (mainly audio, video, and command), (*iii*) reducing the load on the servers, and (*iv*) eliminating the need to install new software on operators devices.

The main contributions of this work can be outlined as follows. First, the related work on *Multiple Operator Single Robot* (*MOSR*) based on the *Web Real-Time Communication* (*WebRTC*) technology is analyzed (Sect. 2). Second, a proof-of-concept of the proposed WebRTC-based telerobotic platform is described (Sect. 3). The developed system presents many interesting features to remote control the whole robot by a single operator or by multiple operators, separately or cooperatively. It offers real-time monitoring of the robot (via video streaming of the workspace) and visualization of data acquired from its sensors. A native WebRTC module is utilized in the robot side enabling direct interaction of operators with its components. The implemented platform allows audiovisual communication and messages exchange between operators while controlling the robot. It also manages the logged-in operators accounts and allows viewing the logs of executed operations/tasks. Finally, the different functionalities of the software platform are illustrated through several validation scenarios (Sect. 4).

2 Related work

One of the principal reasons that pushed researchers to work on MOSR remote control systems is the different studies realized on human-human interactions to jointly realize tasks of objects manipulation. It has been shown that the performance of many humans collaboratively performing such a task is better than that of a single operator accomplishing the same task (Reed and Peshkin 2008).

2.1 MOSR telerobotics over the Internet

A few work only dealt with developing MOSR telerobotic systems over the Internet. A brief survey on the principal research works is given in what follows.

Da Vinci system was developed by Intuitive Surgical in 1999 (Liu et al. 2017); it is a remote operated surgery robot with many 7-DoF manipulators (Germain et al. 2010). This system offers a collaborative control between many operators with a near real-time response (Bodner et al. 2004). However, the robot is accessible via a specific device and the distance between robot and surgeon must not exceed a few meters. *Zeus* is a kind of a master/slave teleoperator between

surgeon and patient-side manipulator (Liu et al. 2017); it has been used in *Lindbergh* operation in September 2001 (Sung and Gill 2001). This system offers a 3D vision and a zoom. Nevertheless, *Zeus* is difficult to use and to understand, and needs specific tools, which require a considerable time for their installation.

Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015) presented a technology that allowed remote groups accessing a shared *PR2* (*Personal Robot 2*) to create remote labs. This solution is made available as open source for other academics. The *OCTOPUS* system has 26-DoF manipulator and offers MOSR control mode (Chen et al. 2017). The robot is controlled collaboratively by two operators. However, the front and back of the flippers are difficult to be precisely controlled by a single operator, at the same time. In addition, the system does not support more than two users, simultaneously. Finally, the operator is not able to control multiple joints, simultaneously.

2.2 WebRTC-based SOSR telerobotic systems

Web Real-Time Communication (*WebRTC*) is relatively a new technology that allows connecting several users at the same time, and enables browser-to-browser communication (Sepulveda 2016). WebRTC is a *JavaScript API* and protocols developed by *W3C* (*World Wide Web Consortium*) and *IETF* (*Internet Engineering Task Force*) (Loreto and Romano 2014). The objective is being to allow Web applications to make use of a *P2P* (Peer-to-Peer) connection for audiovisual calls and chats, without any plugin (Santos-González et al. 2017) and to link applications (VoIP, P2P file sharing) while getting rid of proprietary extension modules, offering a more immediate communication without transmitting the user data to a centralized system (Pinikas et al. 2016). In this case, one or more operators can gain access to the remote teleoperation stream by merely accessing the appropriate Web page in their browser (Tam et al. 2017).

Very limited models deal with WebRTC-based SOSR control of remote robots; besides, most of them only consider mobile robots. Budiharto et al. (2014) presented an obstacle avoidance strategy for intelligent telepresence robot, named as *NUNI*, created particularly for teleconference with many persons. Computer networks are used to teleoperate the robot; therefore, the manager/supervisor at office/industry (using WebRTC) can move the remote robot closer to the desired individual to initiate a conversation/inspection. Other similar works, carried in Cosgun et al. (2013), Budiharto et al. (2013) and Kanigoro et al. (2014), developed a Web application as a part of teleconferencing mobile robot architecture which can serve as a Web conference system. Authors in Pavón-Pulido et al. (2015) presented *Cybi*, which is a companion smart robot. Using a fuzzy-based method, they suggested a distributed software architecture based on connected modules incorporating *Robot Operating System* (*ROS*) (Robot Operating System 2021), WebRTC, and *Google App Engine*. This enables the operator to interact with the robot in a natural and transparent manner. Sundaram et al. (2015) proposed a model architecture for direct control in a remote robot surveillance system via the Internet based on WebRTC. Authors also discussed the implementation and performance of a networked robot system with the help of cloud computing. Kalhapure (2016) proposed an approach based on WebRTC and *MQTT* (*Message Queue Telemetry Transport*) (Message Queue Telemetry Transport 2021; Thangavel et al. 2014) to build a commercial robotic telepresence system called *OATS* (*Open Access Telepresence Systems*). A person can use any smartphone (or a laptop) running an OATS client application to connect and interact with a robotic platform. Another work was done by Ha et al. (2017), who presented an approach to telepresence wheelchair, to real-time video communication and remote interaction, for assisted-living of individuals with impairments. The wheelchair was remotely controlled using a Web browser. Nalamwar et al. (2016) presented *ISAAC* telepresence robot for implementing real-time communication. This system is built using off-the-shelf hardware and open source software modules. Authors also discussed the use of WebRTC technology for video conferencing and remote control of the robotic platform.

2.3 WebRTC-based MOSR systems

WebRTC-based MOSR telerobotic systems were the focus of a few research works.

Tung et al. (2021) presented *Multi-Arm RoboTurk* (*MART*) platform that enables many remote operators to control a robotic arm through low-latency teleoperation, at the same time. The platform uses WebRTC to create communication links between the operator Web browser, smartphone, and the distant server, which interfaces with the robot workspace, to facilitate video streaming to each Web browser. After logging in, six operators will be able to use their smartphones to control the end-effector of the 6-DoF robot. Operators were located at distances ranging from tens to several thousands of miles from the server. Authors validated this approach via a *Transport task* in a shared workspace, and allows pairs of operators to collect data (Mandlekar et al. 2021).

Tan et al. (2019) developed a telepresence robot in a mock-up smart lab to help students to remotely achieve assignments by employing robotics, Internet of Things (IoT) devices, cloud services, virtual reality, and learning analytics. Authors used WebRTC technology for video conferencing to handle multi-party calls. Further, each operator was able to watch the same video from the robot. As well, many users shared the experience of controlling the remote robot via distinct activities. Goga et al. (2021)

presented a low-budget robotic solution to communicate between patients, their families and caregivers in high-risk environments (ICU departments, for example). For this purpose, authors developed a Web platform based on WebRTC technology. The mobile robot is built around a *Raspberry Pi4* that controls an *Arduino*, which in its turn controls the motors. Once connection established, data stream starts, and users will be able to observe and hear each other. Users are also able to rotate/zoom the robot camera, and access the microphone via the Web platform. Therefore, doctors were able to inspect their patients while remaining at a safe distance in their office (for example, during the COVID-19 pandemic).

Yang et al. (2020) developed a telepresence system for remote care in an isolation ward to reduce contact between patients, robot and healthcare staff. Authors used a wearable motion detection device to capture the motion data of the caregiver (using a pair of gloves) and exploited it to control the remote robot. The robot consists of an omnidirectional mobile robot carrying a dual-arm manipulator. The mobile robot is equipped with four Mecanum wheels; the collaborative manipulator robot *YuMi IRB14000* is a 7-DoF two-armed industrial robot. Additionally, authors deployed a multi-users audio/video conference system for remote medical consultation based on WebRTC. They besides created a voice wake-up feature to make patient operation easier. The robot is able to operate remote medical functionalities such as auscultation using a Doppler ultrasound stethoscope, gripping medicine and delivering it to patients (Holland et al. 2021).

2.4 Discussion

Although these telerobotic systems have met the required needs, they present some shortcomings. First, only a small number of operators (six operators, at most) can control the system, simultaneously. Second, it is even impossible for operators to control several joints at the same time. Third, the remote robot is accessed through specific devices. Finally, in most solutions, distance between operators and robots must not exceed several meters (limited distance).

The study of existing telerobotic systems reveals that *MOSR* control requires collaboration between multiple operators for tasks execution. Moreover, the remote robot requires a real-time feedback so that the system can be properly operated. Any delay in communication may cause unfavorable outcomes, which can be hazardous if heavy or fast moving actions are involved (Ishak et al. 2017). These reasons motivated utilizing WebRTC technology, which allows a very important profit for collaborative work solutions.

Unlike other solutions, WebRTC does not require any configuration or additional installations. According to our findings, WebRTC is of a great choice to meet the needs of MOSR telerobotic systems over the Internet (Holmberg et al. 2015). Until recently, WebRTC has been the only standardized technology that allows Web-based P2P bidirectional audiovisual and data communication. It enables exchanging video and audio streams obtained from various sources. WebRTC also provides Web-to-Web audiovisual conferencing without additional plug-ins. Accordingly, for compatibility and ease of integration reasons, telerobotic systems can take advantages of WebRTC technology for yielding remote services and monitoring (Tiberkak et al. 2018).

Table 1 lists the advantages and drawbacks of principal research works studied in this article on *MOSR telerobotics over the Internet*, *WebRTC-based SOSR telerobotic systems* and *WebRTC-based MOSR telerobotic systems*. It should be noted that advantages of WebRTC-based MOSR systems also include those of WebRTC-based SOSR systems.

3 WebRTC-based telerobotic system

This section describes a generic WebRTC-based telerobotic architecture for *MOSR* interaction allowing multiple operators to control the remote robot. To do this, operators are classified according to their access level or priority to control the various sub-systems of the robot. The absolute priority is given to the team leader who assigns the control of the whole or part of the robot to any team member.

3.1 Requirements

A MOSR telerobotic platform should ensure several functionalities, in such a way that operators belonging to a team feel as they are located in the same place near the robot. Moreover, such a platform should enable several operators teams to handle the robot, simultaneously. In addition to obvious criteria when designing telerobotic systems such as security and privacy, a few factors has to be taken into account:

- *Real-time communication*: this is the most important factor; indeed, data must reach destination within a certain period of time. Obviously, the platform delivers the data to the receiver; otherwise, it notifies the sender of the failure.
- *Data*: the platform must be able to transmit a variety of data through the network. This data includes values generated by the robot, sensors, commands sent by operators, feedback provided by the robot, messages exchanged between operators, and any other information that may assist operators to have a full knowledge about the tasks being accomplished.
- *Audio/video*: in addition to raw data, operators should be able to get video and audio streams from the robot as well

Table 1 Advantages and drawbacks of main works on *MOSR over the Internet*, *WebRTC-based SOSR systems* and *WebRTC-based MOSR systems*

Mode	Main works	Advantages	Drawbacks
MOSR over Internet	Liu et al. (2017); Germain et al. (2010); Bodner et al. (2004); Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015); Chen et al. (2017)	Do not require powerful devices	Need specific software, tools and devices Time-consuming installation Small number of operators control the robot, at a time Difficult to use and to understand Lack of real-time feedback Lack of Interoperability Lack of collaboration between operators Do not provide/support a good video quality
WebRTC SOSR	Budiharto et al. (2014); Cosgun et al. (2013); Kanigoro et al. (2014); Pavón-Pulido et al. (2015); Sundaram et al. (2015); Kalhapure (2016); Ha et al. (2017); Nalamwar et al. (2016)	Do not require any configuration or additional installations Unique standardized technology for Web-based P2P real-time bidirectional audio, video and data communication Allow connection even behind NAT servers High-level of interoperability	Require a WebRTC-compatible Web browser Require powerful devices able to run native WebRTC
WebRTC MOSR	Tung et al. (2021); Mandlekar et al. (2021); Tan et al. (2019); Goga et al. (2021); Yang et al. (2020); Holland et al. (2021)	Allow communication between operators Cooperative and concurrent access to the remote robot	Require multiple servers (STUN, TURN...)

as other operators belonging to the same team. Video and audio streams may greatly help operators in obtaining accurate information about the robot, its surroundings, the task progress, operators body language, etc.

- *Cooperative control*: two or more operators may jointly control the robot to performs tasks. In certain circumstances, it is preferred or even required that each operator controls a different part of the robot.
- *Concurrent access*: in some situations, it is expected that the robot or a part of it should be used by one and only one operator at a time, although the other operators are requesting to use it.
- *Anywhere access*: operators must be able to control the robot from any location around the world via an Internet-connected device. All this regardless the use of public, home, or any other kinds of Internet connection.
- *Interoperability*: to the authors best knowledge, the implementation of the *control panel* as a Web page is the optimal option to provide an interoperable means to control the remote robot. This is compatible with any computer-based device including smartphones, tablets and PC. It is not required to add any extension or plugging to Web browsers.
- *Operators communication*: operators should feel themselves in the same location, and perceive themselves to be at the same site as the robot. As a result, the platform must enable real-time communication between operators while also supporting video, audio and data.
- *Multiple teams*: it is preferable to use the same software platform by independent teams. This would optimize the resources usage, whether computational (hardware or software), or human (managing staff).
- *Industrial deployment*: since the platform will be deployed in industrial settings, it must be reliable and user-friendly. In other words, the manager should easily understand how it is implemented and how it operates.

3.2 Proposed WebRTC-based telerobotic system

As Fig. 1 illustrates, the proposed WebRTC-based MOSR remote control of mobile manipulators consists of many teams sharing a set of servers (Web, Signaling, Application, STUN and TURN). Each team is composed of many operators that can control a robot, and a manager that leads the team and administrates their access to the software platform and the robot. All communications between operators, manager and robot are done through the WebRTC connection (audio, video, and data channels). Shared servers are only used while establishing connection, except TURN and application servers. In some circumstances, it is impossible to create direct WebRTC connections; therefore, communications pass through TURN server. However, the application

server can be used during the connection; for example, to store information about tasks accomplished by each operator.

3.3 Proof-of-concept of the proposed system

The developed proof of concept is the deployment of WebRTC technology to control and monitor remote robots. This proof-of-concept allows a team (operators and a manager) to control *RobuTER/ULM* via Internet, simultaneously. For this purpose, the manager defines access rights for each operator. For example, one operator controls the mobile robot, the other controls the manipulator, the third controls the gripper, and another operator controls the whole robot. Besides, a chat system between operators has been developed. Accordingly, operators may exchange real-time text messages and audiovisually communicate to collaboratively control the remote robot.

As Fig. 2 depicts, the proof of concept consists of two servers (web and Signaling), two Web applications (one for operators, one for manager), and a native WebRTC application to ensure communication with the robot. In this implementation, TURN server is not required; instead STUN servers deployed by Google are used. The different components are enumerated as follows:

- *Self-signed certificate*: *Java toolkit keytool* is utilized to create the certificate. This ensures a secure communication between peers and Web server, and between peers and signaling server.
- *Web and Signaling servers*: *Nodejs* is used to implement the Web and the Signaling servers. Both servers use the port 443 and the same IP address to protect them with the same self-signed certificate (because the Web server uses HTTPS protocol and the signaling server uses the secure WebSocket protocol). This avoids importing manually another self-signed certificate to secure WebSocket (which is not a simple operation). Indeed, it is enough to import the certificate for the HTTPS protocol when accessing the Web page.
- *Web applications*: they are implemented in *HTML*, *CSS* and *JavaScript*. Two Web applications are proposed (*i*) one for the operators to control the robot (or a part of it) and communicate between them, (*ii*) the other application for the manager to administrate access rights to the robot.
- *Native WebRTC*: the native WebRTC is implemented using *WebRTC Native API* of the project *webrtc-java* (webrtc-java 2021). It receives operators commands via WebRTC data channels and sends feedbacks from the robot. In additions, it retrieves video stream from the embedded camera and sends it to operators over the video channel. Further, it communicates with the robot via TCP connection, sends data received from WebRTC channel to robot and sends data received from the robot to the corresponding operator via the adequate channel.

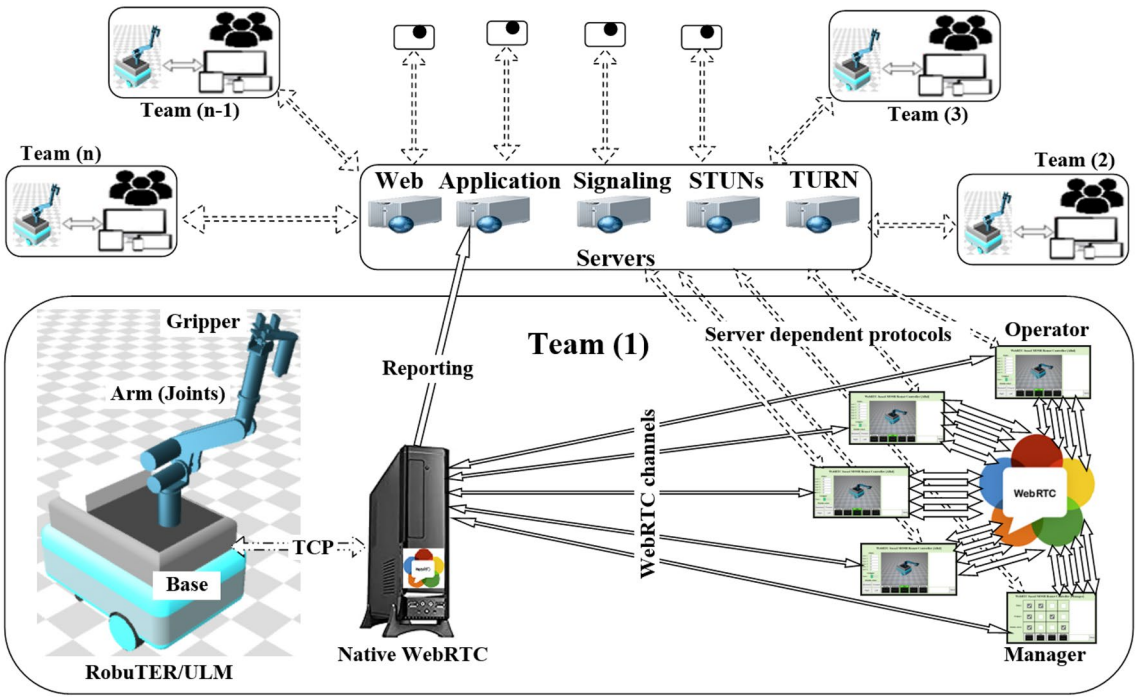


Fig. 1 A synoptic scheme of the proposed WebRTC-based MOSR telerobotic platform

Moreover, it formats data transmitted over data channel in *JSON* format using *JSON.simple* (json-simple 2021), and sends data to application and signaling servers through WebSocket.

- *Application server*: *Java* is used to implement the application server, and *SQLite DBMS* to host the database of the platform. The database contains data about working teams, operators, managers, history of interactions with robots, etc. It is managed by the application server.

Exchanges between operators, manager, and remote robot are made by a direct WebRTC connection between the concerned peers. Regarding this connection, the robot starts first listening on the port 8080 ①; when the native WebRTC application is lunched, it opens a TCP connection to the robot ② to send operator commands and receive feedback (sensors date, etc.), connects to the application server ③ to send reports (for example, report about received commands from operator), and connects to Signaling server to receive connection requests and exchange connection parameters with operators and managers Web applications ④. When an operator gets the Web application from website ⑤ and authenticates successfully ⑥, the Web application sends a connection request to all connected peers (robot, operators belonging to the same team, and the team manager) ⑦. The robot creates a WebRTC connection with video and data channels ⑧; then sends the description of each WebRTC to the corresponding peer through the Signaling server ⑨. Additionally, it retrieves the parameters of each channel (each WebRTC connection has three channels, at most) from STUN servers ⑩ and sends them to the appropriate Web application ⑪. When a Web application receives the native WebRTC connection parameters, it creates a WebRTC connection ⑫ at its side, configures such a connection with received parameters ⑬, and sends the created connection parameters to the native WebRTC (of course, it repeats the same operations with the Web application of the other peers) ⑭. At the same time, it gets the WebRTC channels parameters from the STUN servers ⑮ and sends them to the native WebRTC ⑯ too. When the native WebRTC receives the remote connection parameters, it adds them to the local connection (resp. the other Web applications) ⑰;

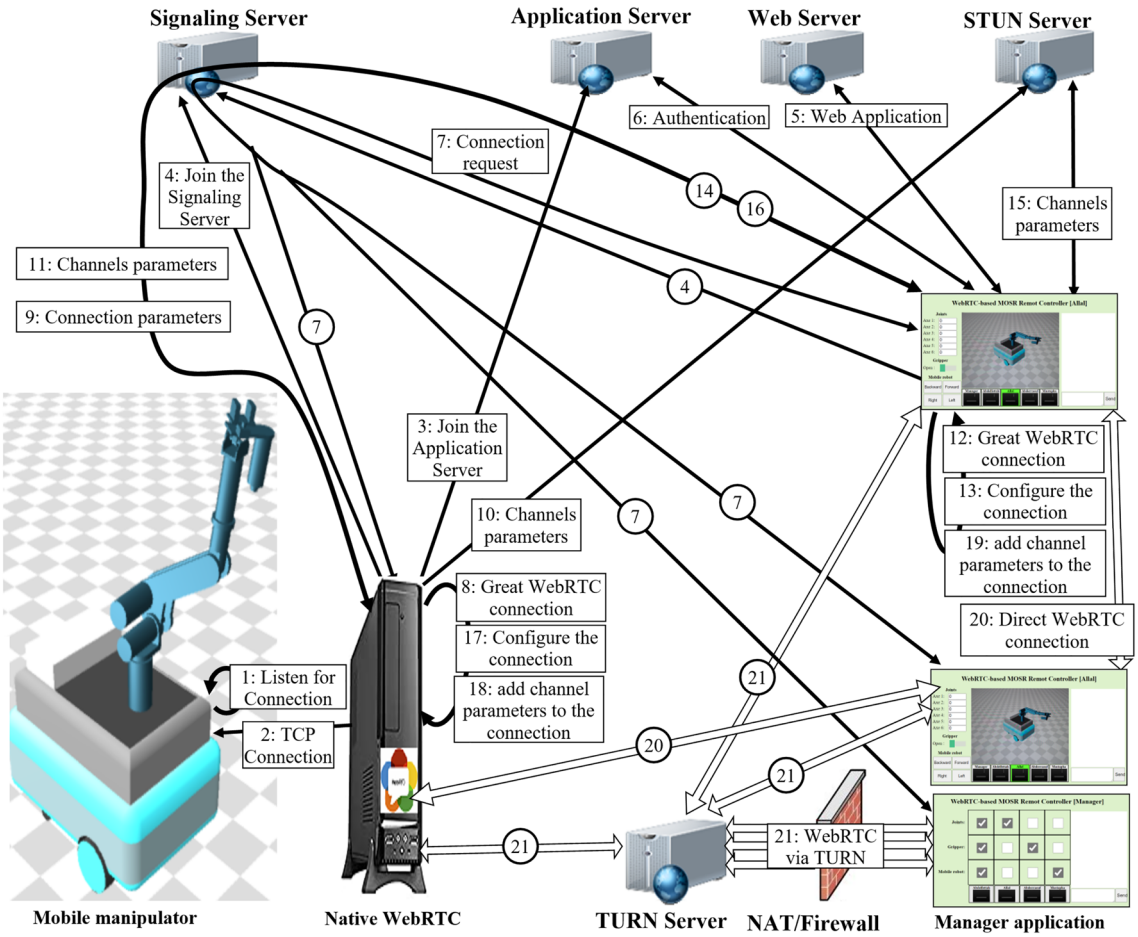


Fig. 2 Proof-of-concept of the proposed WebRTC-based MOSR telerobotic system

besides, when receiving channel parameters, it adds them to the local connection (18). The operator Web application at its turn, when receiving channel parameters, adds them to the corresponding WebRTC connection (19). Depending on the network configuration, WebRTC data channel can directly be created between peers (robot, operators of the team, and manager) (20) or through TURN server (21). In the validation scenario, the TURN server is not used.

4 Validation scenarios

To validate the proposed WebRTC-based MOSR platform for remote control of mobile manipulators, three types of test scenarios have been defined. The first test demonstrates its feasibility; the second assesses its performance; the last test determines the execution times for basic tasks carried by the robot.

Despite the fact that the considered robot is an emulator built in Akli et al. (2010) (not a real robot), this will not affect the obtained results. Indeed, the emulator faithfully imitates all the behaviors of the real mobile manipulator. Moreover, the aim is being to test the developed platform itself not the robot.

4.1 RobuTER/ULM mobile manipulator

Figure 3 shows the available experimental robotic system (Hentout et al. 2013). It consists of a mobile manipulator, *RobuTER/ULM*, controlled by an on-board PC. *RobuTER/ULM* is made up of a 6-DoF Ultra-Light Manipulator (*ULM*) with a two-finger electrical gripper, installed on a rectangular non-holonomic differentially-driven mobile robot (*RobuTER*) (Hentout et al. 2009). The robot features a wireless communication technology to communicate with an off-board PC (powerful enough to run the native WebRTC implementation):

- *RobuTER* has an odometer sensor on each driving wheel, a sick200 laser measurement system in front, a Hokuyo laser sensor at its back and a belt of 24 ultrasound sensors.
- *ULM* has an incremental position sensor on each articulation, a 6-DoF effort sensor as well as a camera installed on its gripper.

4.2 Performance tests

These tests measure the required time to connect Web applications to the signaling server, measuring the time needed to connect Web applications to the native WebRTC gateway, and finally, measuring the round-trip delay between Web application and the emulator of *RobuTER/ULM* through the WebRTC data channel (between Web application and native WebRTC gateway) and the TCP connection (between native WebRTC gateway and emulator).

This study does not consider the bandwidth limits. In fact, the WebRTC seamlessly scales to available bandwidth; when operators notice the degradation of the video/audio quality, they will decide either continue utilizing the software platform or not. When the delay exceeds a certain period, the platform notifies the operators; in this instance, operators will choose to continue working or stop. Indeed, the choice to apply any policy depends on the tasks being realized by the robot, rather than by the robot or the platform.

The platform offers an events management system that tolerates defining its behaviors when unexpected events occur (insufficient bandwidth, important delay, communication lost with an operator or the robot, connection lost between robot and all operators, etc.). In such a case, the proof-of-concept proposes a default behavior, which consists of notifying operations, and moving the robot to a safe position.

The round-trip delay is independent of the task requested by the operator. It is assumed that the native WebRTC gateway is permanently connected to the emulator and signaling server. TCP connection is constantly established between the robot and native WebRTC gateway. As Fig. 4 illustrates:

- *Connection to the signaling server*: it consists of opening a TCP connection to this server, doing TLS handshaking, then opening a WebSocket connection via the opened secure TCP connection. In step (1), the operator Web application opens a secure WebSocket connection to the signaling server.
- *Connection to the robot*: if an operator tries to connect to the robot, the Web application must first connect to the signaling server (1); then, it should send a connection

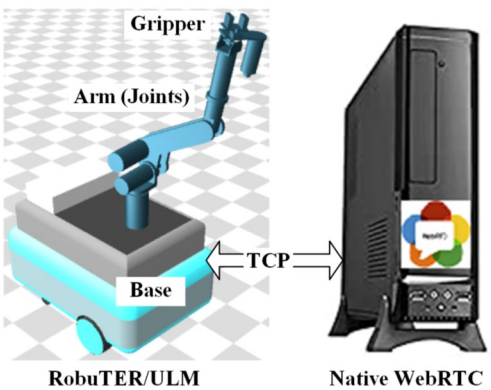


Fig. 3 Available experimental robotic platform

request to the native WebRTC gateway via the signaling server ②. As soon as the native WebRTC gateway receives this request, it creates a WebRTC connection and replies by the parameters of the session of the created connection (session is composed of two channels, one for video and the other for data) ③, it retrieves the networks parameters for each channel from STUN servers ④ and sends them to the operator Web application ⑤. Immediately, When the operator Web application receives these parameters, it creates a local WebRTC connection and generates its session parameters. After that, it sends these parameters to the native WebRTC gateway ⑥, retrieves the network parameters of the channels from the STUN servers ⑦ and sends them to the native WebRTC gateway ⑧. Finally, when the WebRTC data channel is opened, the WebRTC connection to robot is considered as established ⑨.

- *Connection to the robot without being connected to signaling server*: steps from ① to ⑨ must be performed to connect the native WebRTC gateway when the operator is not connected to signaling server.
- *Round-trip delay*: the round-trip delay is the required time for a data to travel from the Web application to the robot and to return back to the Web application through the native WebRTC gateway. The packets holding data are sent from the Web application to the native WebRTC gateway via the webRTC data channel ⑩; this latter sends the received packets from the Web application to the robot via the TCP connection ⑪. As soon as a packet is received, the robot sends it back to the native WebRTC gateway via the TCP connection ⑫. Finally, the native WebRTC gateway sends the received packet from the TCP connection to the Web application via the WebRTC data channel ⑬. The robot should inform the operator immediately, whether the request is taken into account or not.

As Fig. 5 illustrates, the test platform of the proof-of-concept consists of the following components:

- *Laptop*: Dell Inspiron I7548, Intel Core i7-5500U 2.4GHz processor (two cores, four threads), RAM 16GB (DDR3-SDRAM) and 512GB (SSD). It runs 64-bit Windows 10 Home edition, Netbeans 8.2, jdk1.8.0_101, sqlite-jdbc−3.36.0, the robot emulator, Microsoft Edge Version 95.0.1020.30, and the implemented servers (Signaling, Web and Application).
- *Smartphone*: Huawei P20 EML-L29, Octa-Core, 2 processors (2.36Ghz Quad-Core ARM Cortex-A73 and 1.8Ghz Quad-Core ARM Cortex-A53), RAM 4GB (LPDDR4) and 128GB internal storage. It runs Android 10, and Chrome version 95.0.4638.50.
- *ZTE modem router*: LAN info (Wi-Fi), and WAN info (Interface ppp0, Type PPPoE, NAT Enabled, Firewall Enabled, and 10Mb/s ADSL subscription).

Tests have been accomplished in three distinguished cases. Further, each test is repeated 30 *times*; the duration between two successive runs is 60*s* to make them independent.

- *Local host*: all the platform components run on the same machine (a laptop).
- *Local network*: the operator Web application is running on the same network as the other platform components. The operator Web application is running on a smartphone *P*20 (node *A*); the other components run on the laptop. The smartphone and the laptop are connected via the ZTE modem router.
- *Internet*: the smartphone that runs the operator Web application is connected to the laptop via the Internet (ADSL on laptop side and 3G++ on smartphone side).

Results, given in milliseconds (*ms*), are presented in Table 2 (DL: Delay; WS: WebSocket; WR: WebRTC; WSR: WebSocket-WebRTC). As illustrated, some values are considered

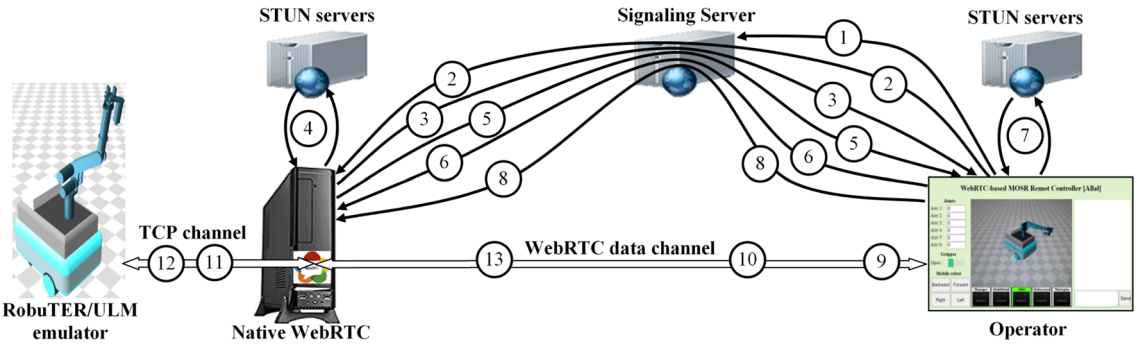


Fig. 4 The proposed and implemented test scenario

as outliers because they are negative or very big compared to the other ones (the value is three times greater than the average); they have been replaced by the average of the other values of the same test.

Before discussing the results, some information about the delays in the field of telerobotics is indicated by Farajiparvar et al. (2020) as follows:

- A one-way delay which is less than 250 ms is considered as a non-delay (a round-trip delay is less than 500 ms).
- A one-way delay which is less than 400 ms is tolerable (a round-trip delay is less than 800 ms).
- Even if the one-way delay is greater than 400 ms (a round-trip delay above 800 ms), it is possible to realize remote operations with less performance.

Table 3 presents the maximum, the average and the ratio of the maximum to the average of the round-trip and connection times for each test of Table 2. Table 3 also shows the number of round-trip delay values, which are less than 500 ms (considered as non-delays), those belonging to [500 ms, 800 ms] (considered as tolerable), and values greater than 800 ms (considered as delays).

4.2.1 Local host

The sum of maximum time of WebRTC data channel, maximum connection time to signaling server added to that of WebRTC is equal to $7 + 77 + 230 = 314$ ms. In the worst case, this sum is less than 500 ms; this means that the communication delay between the Web applications of operators and the robot is considered as a non-delay. Further, the WebRTC data channel time is equal to 304 ms only once (second line in Table 2). This delay is much bigger than the average value; as a consequence, it is considered as outlier value and is not taken into account in the evaluation of this proof-of-concept.

4.2.2 Local network

The sum of maximum time of WebRTC data channel and that of WebRTC connection is equal to $24 + 343 = 367$ ms. Since it is less than 500 ms, this means that it is considered as a non-delay. In case a WebRTC disconnection occurs, both peers (operator Web application and native WebRTC module) remain connected to the signaling server.

In case of disconnection from both Signaling server and the WebRTC, the sum of maximum time of WebRTC data channel, the maximum connection time to the signaling server, and that of WebRTC connection is $24 + 343 + 427 = 794$ ms. Since it is less than 800 ms, this indicates that the delay is reasonable. Table 2 shows that 90% (27 of 30) of the sum of connection time to signaling server and that of WebRTC connection time are less than $500 - 24 = 476$ ms (24 ms is the maximum value of the round-trip delay); this means that 90% of cases are considered as a non-delay.

However, in three cases, the round-trip delay is greater than 800*ms*. These delays are three times greater than the maximum delay. Such values are considered as outliers; as a consequence, they are not taken into account in the evaluation of this proof-of-concept.

4.2.3 Internet

As Table 2 illustrates, 86.76% (26 amongst 30) of measured round-trip latencies are less than 500*ms*; this means that

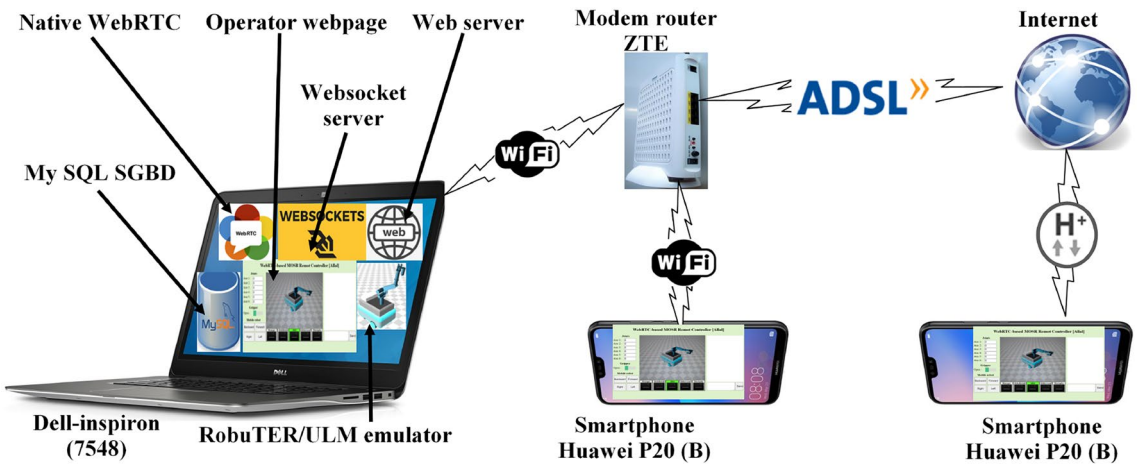


Fig. 5 Test platform of the developed proof-of-concept

Table 2 Measured values of round-trip and connection delays

	Local host				Local network				Internet			
i	DL	WS	WR	WSR	DL	WS	WR	WSR	DL	WS	WR	WSR
1	6	51	129	180	~~218~~ **12.96**	134	299	433	389	792	2709	3501
2	~~304~~ **3.34**	60	152	212	~~82~~ **12.96**	140	289	429	538	935	2393	3328
3	3	59	206	265	14	343	315	658	404	763	2453	3216
4	3	77	181	258	12	150	294	444	410	860	2399	3259
5	4	50	132	182	8	172	290	462	389	808	2424	3232
6	4	55	146	201	~~102~~ **12.96**	170	393	563	406	733	2415	3148
7	3	67	166	233	10	173	310	483	436	801	2444	3245
8	2	65	140	205	9	148	413	561	394	878	2472	3350
9	3	63	128	191	9	134	276	410	400	~~4351~~ **857.92**	2438	6789
10	3	56	124	180	24	146	282	428	388	734	2485	3219
11	3	46	130	176	10	138	265	403	385	727	2423	3150
12	3	50	130	180	18	152	287	439	396	900	2479	3379
13	2	47	147	194	12	136	285	421	~~16207~~ **409.74**	919	2374	3293
14	2	57	129	186	17	154	307	461	383	~~-18290~~ **857.92**	~~19794~~ **2498.04**	~~1504~~ **3397.46**
15	2	49	186	235	13	158	427	585	382	2172	3272	5444
16	3	61	123	184	8	139	282	421	381	619	2422	3041
17	2	54	133	187	9	132	238	370	390	670	2443	3113
18	7	51	128	179	17	140	291	431	381	~~7415~~ **857.92**	2977	~~10392~~ **3397.46**
19	3	61	125	186	12	144	278	422	397	827	2467	3294
20	3	67	133	200	9	133	259	392	381	799	2610	3409
21	6	74	230	304	20	132	215	347	388	804	2496	3300
22	3	67	125	192	12	125	243	368	382	1192	2438	3630
23	4	76	118	194	9	173	267	440	~~7703~~ **409.74**	896	2382	3278
24	3	61	126	187	15	245	313	558	382	782	2438	3220
25	2	68	124	192	23	242	385	627	731	846	2394	3240
26	3	48	133	181	17	118	252	370	~~7496~~ **409.74**	921	2480	3401
27	4	52	133	185	10	130	208	338	386	683	2469	3152
28	4	64	126	190	13	134	228	362	385	625	2390	3015
29	4	47	130	177	9	125	239	364	385	620	2359	2979
30	3	48	132	180	11	152	206	358	394	~~11150~~ **857.92**	~~15138~~ **2498.04**	~~26288~~ **3397.46**

Bold strikethrough numbers represent outliers

Bold numbers represent the average of the other values of the column

they are considered as a non-delay. Three of these values are very big and are treated as outliers (they do not affect the evaluation of this proof-of-concept). The fourth value above 500 ms is equal to 538 ms. This latency is less than 800 ms, which makes it acceptable for telerobotic applications.

It must be noted that establishing connection to the signaling server and to the native WebRTC module took too much more time than 800 ms. Thus, it is very important to keep all peers (Web applications and native WebRTC module) connected to signaling server and reestablish this connection as soon as disconnected (to reduce costs in case of a disconnection of operators and/or manager Web applications from the native WebRTC module).

4.3 Feasibility test

The feasibility test is done on the same machine (a laptop) that runs the software platform. It consists of executing the emulator, servers (Signaling, Application, and Web), and the native WebRTC gateway; finally, opening several tabs in the browser, one for the manager and the other ones for the operators.

Table 3 Maximum, average, number of non-delay, tolerable and delay values of round-trip and connection times

	Local host				Local network				Internet			
	DL	WS	WR	WSR	DL	WS	WR	WSR	DL	WS	WR	WSR
MAX	7.00	77.00	230.00	304.00	24.00	343.00	427.00	658.00	731.00	1192.00	3272.00	6789.00
AVR	3.34	58.37	141.50	199.87	12.96	157.07	287.87	444.93	409.74	805.36	2498.04	3397.46
$\frac{MAX}{AVG}$	2.09	1.32	1.63	1.52	1.85	2.18	1.48	1.48	1.78	2.53	1.31	2.00
< 500	29	–	–	–	27	–	–	–	26	–	–	–
[500, 800]	1	–	–	–	3	–	–	–	4	–	–	–
> 800	1	–	–	–	3	–	–	–	3	–	–	–

Figure 6a shows the Web page of operators after authentication. It includes a lateral panel with various options for controlling the robot at the left side, a chat system at the right side, video delivered by the eye-to-hand camera (visualizing the robot inside its workspace) at the center, and videos of the other operators of the team at the far right of the Web page. Further, the panel contains options to control the mobile robot velocity and orientation (Backward, Forward, Left, Right). Finally, this page offers the possibility to control the 6-DoF manipulator robot ($Axe_1 \ldots Axe_6$), and to open/close its gripper.

Figure 6b shows the Web page of the manager where he can define the access rights for each operator. For example, *Abdelfetah* can control the whole robot (mobile robot, 6-DoF manipulator, and gripper); *Allal* is able to control the 6-DoF manipulator only.

4.4 Execution times

The validation of the proposed WebRTC telerobotic system is carried out through the same scenarios established in (Khiter et al. 2012). Initial and final conditions are given as shown in Table 4; positions are given in millimeters (mm), orientations in degrees (°) and execution times in seconds (s).

- $\text{Base}_{\text{Init}}(x_B, y_B, \theta_B)_{\text{Init}}$: initial pose of the mobile robot.
- $\text{Base}_{\text{Fin}}(x_B, y_B, \theta_B)_{\text{Fin}}$: final pose of the mobile robot.
- $\text{Effector}_{\text{Init}}(x_E, y_E, z_E, \psi_E, \theta_E, \phi_E)_{\text{Init}}$: initial pose of the end-effector.
- $\text{Configuration}_{\text{Init}}(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_{\text{Init}}$: initial configuration of the arm corresponding to $Effector_{Init}$.
- $\text{Effector}_{\text{Fin}}(x_E, y_E, z_E, \psi_E, \theta_E, \phi_E)_{\text{Init}}$: final pose of the end-effector.
- $\text{Configuration}_{\text{Fin}}$: final configuration of the manipulator corresponding to $\text{Effector}_{\text{Fin}}$; it is calculated using the Inverse Kinematic Model (IKM) of the manipulator.

In Khiter et al. (2012), authors defined four basic types of tasks to be performed by the remote robot (Table 4):

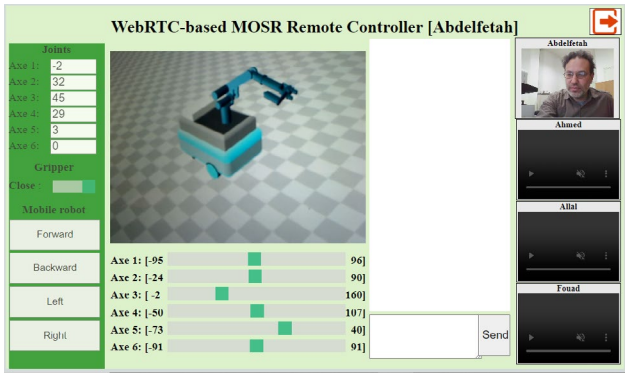
- *First task* (T_1): moving the manipulator from $\text{Effector}_{\text{Init}}$ to $\text{Effector}_{\text{Fin}}$.
- *Second task* (T_2): moving the manipulator from its $\text{Configuration}_{\text{Init}}$ successively to three different configurations: Configuration_1, Configuration_2 and $\text{Configuration}_{\text{Fin}}$.
- *Third task* (T_3): moving the mobile robot from $\text{Base}_{\text{Init}}$ to Base_{Fin} in presence of one obstacle at position $\text{Obstacle}_3(x_3, y_3, z_3)$ of a size Size_3.
- *Fourth task* (T_4): moving the mobile robot from $\text{Base}_{\text{Init}}$ to Base_{Fin}. Two obstacles are present inside the workspace. The first obstacle is at position $Obstacle_{41}(x_{41}, y_{41}, z_{41})$ of a size Size_{41}; the second is at position $\text{Obstacle}_{42}(x_{42}, y_{42}, z_{42})$ of a size of about Size_{42}.

For each basic task (T_1, T_2, T_3, T_4), 10 different runs have been achieved by the robot for the three previous cases (*Local host*, *Local network* and *Internet*). Figure 7 shows the average execution times.

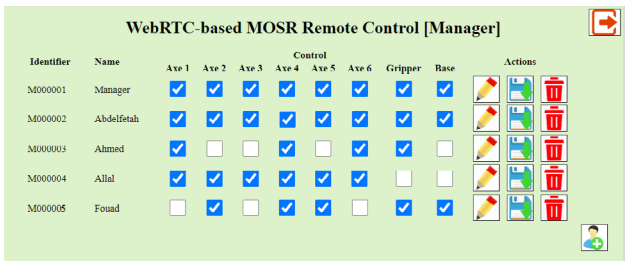
4.5 Management of multiple operators

The considered validation scenario consists of moving *Object* from its initial position *Source* toward a final predefined position *Target* by the remote robot. It is clear that this task is of a high level and should thus be decomposed into primitive operations (Hentout et al. 2010). Therefore, it is considered as composed of three primitive operations (*i*) moving the mobile robot, (*ii*) moving the manipulator robot and (*iii*) localizing *Object* and calculating its coordinates. Two different cases are considered:

- *Single user*: only one user controls remotely the whole robot (mobile robot sub-system, manipulator robot sub-system, and on-board vision sub-system) to perform this task.



(a) Web page of the *Operators*



(b) Web page of the *Manager*

Fig. 6 The Web applications to control the remote robot and to manage the users

- *Multiple users*: three users collaboratively control the remote robot. For example, the first user controls the motion of the mobile robot. At the same time, the second user configures the manipulator robot to grasp/deposit *Object*. The last user controls the on-board camera to localize the corresponding *Object*. After that, the first user moves the robot to the final position where the second user will deposit the grasped *Object*.

The team manager may assign tasks to operators based on their location, for example. In this case:

- *Control of the gripper*: it is the most critical task. It is assigned to operator connected directly to the robot where he can monitor the robot without using the camera. However, the operator must use the platform to be able to cooperate with the other users.
- *Control of the manipulator joints*: it is less critical than that of the gripper. This task is assigned to an operator connected to the robot via a local network.
- *Control of the mobile robot*: it is the least critical. Indeed, imprecision on manipulating this sub-system can be recovered by the joints controller; thus, it can be assigned to any operator connected via the local network or Internet.

The *single operator* carries the predefined task in 10*s*; whereas, the *three collaborative operators* were able to perform the same task in 03*s*.

4.6 Comparison with solutions of the literature

Table 5 gives comparison between the proposed WebRTC-based solution with those of the literature on many criteria described in subsection 3.1.

The proposed WebRTC-based MOSR telerobotic platform of mobile manipulators meets all these criteria. In fact, it enables real-time audio (unlike Germain et al. (2010); Bodner et al. (2004); Chen et al. (2017); Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015); Sundaram et al. (2015); Goga et al. (2021)), video (unlike Liu et al. (2017)) and data communication between operators, and between the robot and operators. Moreover, the platform allows operators to directly reach the other ones and the remote robot regardless of their locations, even if their connected devices are behind NAT servers (unlike Liu et al.

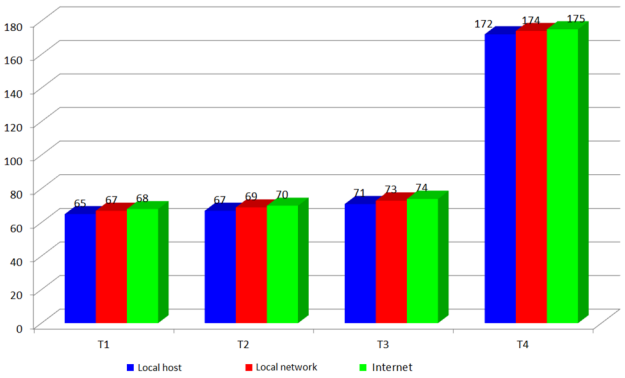


Fig. 7 Average execution times for the considered tasks (T_1, T_2, T_3, T_4) in all cases

Table 4 Initial and final conditions of the considered validation tasks

	Parameter	Value
Initial conditions	$\text{Base}_{\text{Init}}(x_B, y_B, \theta_B)_{\text{Init}}$	$(0, 0, 0°)$
	$\text{Configuration}_{\text{Init}}(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_{\text{Init}}$	$(0, 0, 0, 0, 0, 0)$
	$\text{Effector}_{\text{Init}}(x_E, y_E, z_E, \psi_E, \theta_E, \phi_E)_{\text{Init}}$	$(-432, -108.49, 164, -180°, -180°, -180°)$
Task T_1	$\text{Effector}_{\text{Fin}}(x_E, y_E, z_E, \psi_E, \theta_E, \phi_E)_{\text{Init}}$	$(-330mm, -630mm, 1080mm, -135°, -88°, 5°)$
	$\text{Configuration}_{\text{Fin}}(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_{\text{Fin}}$	$(-60°, 61°, 30°, 95°, -15°, 0°)$
Task T_2	$\text{Configuration}_1(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_1$	$(0°, 40°, 28°, 0°, 0°, 0°)$
	$\text{Configuration}_2(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_2$	$(20°, 32°, 28°, 0°, 0°, 0°)$
	$\text{Configuration}_{\text{Fin}}(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6)_{\text{Fin}}$	$(0°, 45°, 45°, 0°, 0°, 0°)$
Task T_3	$\text{Base}_{\text{Fin}}(x_B, y_B, \theta_B)_{\text{Fin}}$	$(-1920, 2, 15°)$
	$\text{Obstacle}_3(x_3, y_3, z_3)$	$(-1000, 0, 50)$
	Size_3	$(800 \times 200 \times 100)$ mm
Task T_4	$\text{Base}_{\text{Fin}}(x_B, y_B, \theta_B)_{\text{Fin}}$	$(-3440, 13, 12°)$
	$\text{Obstacle}_{41}(x_{41}, y_{41}, z_{41})$	$(-1000, 400, 50)$
	Size_{41}	$(800 \times 200 \times 100)$ mm
	$\text{Obstacle}_{42}(x_{42}, y_{42}, z_{42})$	$(-2000, -400, 50)$
	Size_{42}	$(600 \times 250 \times 100)$ mm

(2017); Germain et al. (2010); Bodner et al. (2004)). Further, even it requires using a set of servers (Web, Signalling and perhaps a STUN server), they are only used when joining the platform; thus, the latter could be used by a high number of operators organized in multiple teams. All the other solutions do not assure this requirement except that developed in Tung et al. (2021). Regarding the concurrent and cooperative access to the remote robot, our solution implements both requirements. On the one hand, those presented in Liu et al. (2017); Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015); Pavón-Pulido et al. (2015); Kanigoro et al. (2014); Sundaram et al. (2015); Kalhapure (2016); Ha et al. (2017); Budiharto et al. (2014); Goga et al. (2021); Yang et al. (2020) do not verify the cooperative control requirement. On the other hand, among all the considered solutions for comparison, only Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015); Goga et al. (2021); Tung et al. (2021) meet the concurrent access requirement. Furthermore, since our solution is based on Web technology, the interoperability is guaranteed unlike those designed in Liu et al. (2017); Germain et al. (2010); Bodner et al. (2004); Chen et al. (2017); Kreczmer et al. (2015). Finally, the deployment of WebRTC-based platform is easy because it only requires using a reduced set of *JavaScript API*. Indeed, all the complex underlying technologies such as communication protocols, security protocols and audio/video coder/decoder are handled by the Web browser; this significantly makes the platform usable in industrial settings in contrast to Pavón-Pulido et al. (2015); Kanigoro et al. (2014); Kreczmer et al. (2015); Sundaram et al. (2015); Kalhapure (2016); Ha et al. (2017); Budiharto et al. (2014); Goga et al. (2021); Tung et al. (2021); Tan et al. (2019).

4.7 Discussion of obtained results

This work proposed a WebRTC-based MOSR remote control platform for mobile manipulators. It allows multiple operators to collaboratively control a single remote robot by using Web applications. Operators can use any Web browser to control the remote robot without any pre-configuration or additional installation, neither on the Web browser, operating system nor any hardware/software of the machine running the Web browser. Other major advantages offered by the developed platform may be listed as follows. First, this platform allows an operators team to control the remote robot located anywhere by simply connecting to Internet via mobile networks or just be connected to the same IP network. Second, operators can play real-time or recorded videos/audios from the robot station. Third, operators can communicate with each other via audiovisual or by using textual messages. Finally, many teams can use simultaneously the platform to control their robots.

In the aim of validating the developed platform, a proof of concept have been proposed. It enables a team composed of many operators and a manager to control the remote robot by using Web applications. Each member can display real-time video captured by the robot. Besides, each operator is able to watch other team members, discuss and exchange messages with them. Finally, the Web application enables operators to visualize the robot workspace by displaying the video of the on-board camera.

Once connected to the Web interface through a browser, the operator has to authenticate to access the different subsystems of the robot depending on the privilege set up by the manager. Tasks are of five categories (*i*) control of the robot joints only, (*ii*) control of the robot gripper only, *(iii)* control of the mobile robot only, or (*iv*) control of the whole robot.

Many validation scenarios have been performed including connection tests, feasibility tests, and cooperative control tests. The connection to the remote robot has been tested in three cases (*i*) *direct connection in the same local host*, (*ii*) *connection through the local network* and, finally, (*iii*) *connection via the Internet*. It is evident that connecting through the Internet requires significantly longer time than connecting via the *local host* or the *local network*. To show the superiority of the proposed WebRTC solution, obtained results have been compared with those mentioned in Khiter et al. (2012).

The developed WebRTC-based solution has globally demonstrated a better gain in terms of *transmission time* and *commands admissibility*. The proof-of-concept offers operators of the Web applications accessibility via the Internet to control the remote robot. Finally, results clearly showed that when three operators collaborate to perform complex tasks, the work is accomplished much quicker than one operator.

5 Conclusions and future work

This paper described a generic WebRTC-based platform for remote control of mobile manipulators. The developed system facilitates the utilization of the remote robots to accomplish complex tasks via the Internet while exploiting WebRTC technology. Through the Web interface, operators belonging to the same team have a panel to control part/whole of their robot. They also have control mechanisms of the mobility, manipulation, end-effector and embedded sensors. The Web interface allows operators to display sensors data and visualize the robot workspace using the videos acquired by the on-board camera. Despite the fact that the platform is based on Web technology, the latency is acceptable for remote control of robots through the Internet. Generally, the round-trip delay is less than 500 ms and less than 800 ms in worst cases. While performing primitive tasks, a difference of less than 3 s (1–3 s much more) in the

Table 5 Comparison of the proposed WebRTC-based MOSR platform with the studied works of the literature

Approach	Real time	Audio	Video	Cooperative control	Concurrent access	Inter-operability (Terminals)	Anywhere access	Multiple teams	Operators communication	Industrial deployment
Our solution	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Liu et al. (2017)	Yes	Yes	No	No	No	No	No	No	No	Yes
Germain et al. (2010)	Yes	No	Yes	Yes	No	No	No	No	No	Yes
Bodner et al. (2004)	Yes	No	Yes	Yes	No	No	No	No	No	Yes
Osentoski et al. (2012); Pitzer et al. (2012); Toris et al. (2015)	Yes	No	Yes	No	Yes	Yes	Yes	No	No	Yes
Chen et al. (2017)	Yes	No	Yes	Yes	No	No	Yes	No	Yes (same room)	Yes
Budiharto et al. (2014)	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No
Kanigoro et al. (2014)	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	No
Pavón-Pulido et al. (2015)	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No
Sundaram et al. (2015)	Yes	No	Yes	No	No	Yes	Yes	No	No	No
Kalhapure (2016)	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No
Ha et al. (2017)	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No
Nalamwar et al. (2016)	Yes	No	Yes	Yes	No	Yes	Yes	No	No	Yes
Tung et al. (2021)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Tan et al. (2019)	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No
Goga et al. (2021)	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No
Yang et al. (2020)	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Kreczmer et al. (2015)	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	No

average execution times is noticed for all tasks. In light of all the other advantages of using the Internet for telerobotics, this augmentation seems acceptable. Future work aims at enhancing the developed platform by reducing the consumed bandwidth by acquired videos; consequently, reducing battery consumption in case of using mobile devices with wireless connection. In order to improve overall performances, an algorithm will be developed to estimate the timeout to deal with delayed connection messages to signaling server, to the robot, and of sending commands to the robot by the Web applications. Finally, the developed WebRTC-based MOSR remote control platform will be validated in a real industrial environment by using the real mobile manipulator robot (RobuTER/ULM).

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aghili, H.: The remote control a tele-robot 5R by using dual-tone modulation. In: Applied Mechanics and Materials, vol. 307, pp. 103–106 (2013). Trans Tech Publications

Akli, I., Hentout, A., Bouzouia, B., Daoud, S.: Design and development of mobile manipulator simulator. Application: The RobuTER/ULM mobile manipulator. In: The International Conference on Modeling, Simulation and Control (ICMSC2010), IEEE, pp. 370–374 (2010)

Bodner, J., Wykypiel, H., Wetscher, G., Schmid, T.: First experiences with the da vinci^{TM} operating robot in thoracic surgery. Eur J Cardio-Thorac Surg **25**(5), 844–851 (2004)

Budiharto, W., Moniaga, J., Aulia, M., Aulia, A.: A framework for obstacles avoidance of humanoid robot using stereo vision. Int. J. Adv. Robot. Syst. **10**(4), 204 (2013)

Budiharto, W., Kanigoro, B., Ohyver, M., Shodiq, M., Nugraheni, C., Lim, R., Wicaksono, H.: Obstacles avoidance for intelligent telepresence robot using interval Type-2 FLC. ICIC Express Lett. **8**(3), 1–7 (2014)

Burgard, W., Cremers, A.B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: The interactive museum tour-guide robot. In: AAAI/IAAI, pp. 11–18 (1998)

Carelli, R., Forte, G., Canali, L., Mut, V., Araguas, G., Destefanis, E.: Autonomous and teleoperation control of a mobile robot. Mechatronics **18**(4), 187–194 (2008)

Chen, K., Kamezaki, M., Katano, T., Kaneko, T., Azuma, K., Seki, M., Ichiryu, K., Ishida, T., Sugano, S.: Usability test in different types of control-authority allocations for multi-operator

single-robot system octopus. In: The International Conference on Applied Human Factors and Ergonomics. Springer, pp. 675–685 (2017)
Cloosterman, M.B., Van de Wouw, N., Heemels, W., Nijmeijer, H.: Stability of networked control systems with uncertain time-varying delays. IEEE Trans. Autom. Control **54**(7), 1575–1580 (2009)
Cosgun, A., Florencio, D.A., Christensen, H.I.: Autonomous person following for telepresence robots. In: The International Conference on Robotics and Automation (ICRA'13), IEEE, pp. 4335–4342 (2013)
Crainic, M.-F., Preitl, S.: Virtual laboratory for a remotely operating robot arm. In: The 9th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 101–104 (2014). IEEE
Dalton, B.: A framework for internet robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98). Workshop Robots on the Web, pp. 15–21. IEEE, Victoria, Canada (1998)
Dalton, B.: Techniques for web telerobotics. PhD thesis, University of Western Australia (2001)
Elkady, A., Sobh, T.: Web-based control of mobile manipulation platforms via sensor fusion. In: Web-based control and robotics education, pp. 297–312. Springer, Dordrecht (2009)
Emharraf, M., Saber, M., Rahmoun, M., Azizi, M.: Control architecture for mobile robot teleoperation. In: Proceedings of the Mediterranean Conference on Information & Communication Technologies, pp. 687–692. Springer (2016)
Farajiparvar, P., Ying, H., Pandya, A.: A brief survey of telerobotic time delay mitigation. Front. Robot. AI **7** (2020)
Flückiger, L., Baur, C., Clavel, R., Schweitzer, G., Siegwart, R., Cattin, P.: Cinegen: A rapid prototyping tool for robot manipulators. In: The 4th International Conference on Motion and Vibration Control (MOVIC'98), vol. 1, pp. 129–134 (1998)
Germain, M., Liverneaux, P., Missana, M.-C.: Microchirurgie avec le robot da vinci s. la télémicrochirurgie: l'essor imminent. E-mémoires de l'Académie Nationale de Chirurgie, Paris 9, 74–77 (2010)
Goga, N., Radu, M.D., Vasilăţeanu, A., Păvăloiu, B., Hang, A., Popa, R., Trocmaer, A., Scurtu, D.: Interconnected web platform with autonomous robots for helping patients in ICU sections. In: The 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), IEEE, pp. 1–5 (2021)
Goldberg, K., Song, D., Khor, Y., Pescovitz, D., Levandowski, A., Himmelstein, J., Shih, J., Ho, A., Paulos, E., Donath, J.: Collaborative online teleoperation with spatial dynamic voting and a human "tele-actor". In: The International Conference on Robotics and Automation (ICRA'02), vol. 2, IEEE, pp. 1179–1184 (2002)
Grange, S., Fong, T., Baur, C.: Effective vehicle teleoperation on the world wide web. In: Proceedings of the International Conference on Robotics and Automation (ICRA'00), vol. 2, IEEE, pp. 2007–2012 (2000)
Ha, V.K.L., Chai, R., Nguyen, H.T.: Real-time WebRTC-based design for a telepresence wheelchair. In: Engineering in Medicine and Biology Society (EMBC), The 39th Annual International Conference of the IEEE, pp. 2676–2679 (2017)
Hentout, A., Bouzouia, B., Toukal, Z., Toumi, R.: Multi-agent remote control of the RobuTER/ULM mobile manipulator robot. In: The International Conference on Mechatronics (ICM 2009), IEEE, pp. 1–6 (2009)
Hentout, A., Bouzouia, B., Toumi, R.: Multi-agent missions planning for mobile manipulators. In: The International Conference on Robotics and Biomimetics (ROBIO), IEEE, pp. 1037–1042 (2010)
Hentout, A., Benbouali, M., Akli, I., Bouzouia, B., Melkou, L.: A telerobotic human/robot interface for mobile manipulators: a study of human operator performance. In: The International Conference on Control, Decision and Information Technologies (CoDIT'13), IEEE, pp. 641–646 (2013)
Holland, J., Kingston, L., McCarthy, C., Armstrong, E., O'Dwyer, P., Merz, F., McConnell, M.: Service robots in the healthcare sector. Robotics **10**(1), 47 (2021)
Holmberg, C., Hakansson, S., Eriksson, G.: Web real-time communication use cases and requirements. Technical report, IETF, (2015)
Ishak, M.K., Kit, N.M.: Design and implementation of robot assisted surgery based on Internet of Things (IoT). In: The International Conference on Advanced Computing and Applications (ACOMP), IEEE, pp. 65–70 (2017)
Jara, C.A., Candelas, F.A., Puente, S.T., Torres, F.: Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. Comput. Educ. **57**(4), 2451–2461 (2011)
json-simple (2021). https://github.com/fangyidong/json-simple
Kalhapure, S.: OATS: An architectural design and implementation for telepresence robots. Int. J. Comput. Appl. **150**(12) (2016)
Kanigoro, B., Budiharto, W., Moniaga, J.V., Shodiq, M.: Web based conference system for intelligence telepresence robot: a framework. J. Comput. Sci. **10**(1), 10 (2014)
Khiter, B., Hentout, A., Boutellaa, E., Benbouali, M., Bouzouia, B.: Internet-based telerobotics of mobile manipulators: Application on RobuTER/ULM. In: International Conference on Intelligent Robotics and Applications, Springer, pp. 635–644 (2012)
Kreczmer, B., Grzeszczak, F., Szczesniak-Stanczyk, D., Arent, K., Stanczyk, B.: Video conferencing applications for ReMeDi robotic system. J. Med. Imaging Health Inform. **5**(8), 1622–1630 (2015)
Liu, H.-H., Li, L.-J., Shi, B., Xu, C.-W., Luo, E.: Robotic surgical systems in maxillofacial surgery: a review. Int J. Oral Sci. **9**(2), 63 (2017)
Loreto, S., Romano, S.P.: Real-time communication with WebRTC: peer-to-peer in the browser. O'Reilly Media Inc, California (2014)
Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., Martín-Martín, R.: What matters in learning from offline human demonstrations for robot manipulation. In: The 5th Annual Conference on Robot Learning (CoRL2021), pp. 1–36 (2021)
Message Queue Telemetry Transport (2021). http://mqtt.org
Mostefa, M., El Boudadi, L.K., Loukil, A., Mohamed, K., Amine, D.: Design of mobile robot teleoperation system based on virtual reality. In: The 3rd International Conference on Control, Engineering & Information Technology (CEIT), IEEE, pp. 1–6 (2015)
Moutaouakkil, F., El Bakkali, M., Medromi, H.: New approach of telerobotic over internet. In: Proceedings of the World Congress on Engineering and Computer Science, vol. 1, pp. 20–22 (2010)
Nalamwar, S., Kalhapure, S., Khatake, A., Gandhi, S., Jain, K.: Real time communication using embedded system beyond videoconferencing and towards telepresence. Int. J. Comput. Appl. **134**(14) (2016)
Osentoski, S., Pitzer, B., Crick, C., Jay, G., Dong, S., Grollman, D., Suay, H.B., Jenkins, O.C.: Remote robotic laboratories for learning from demonstration. Int. J. Soc. Robot. **4**(4), 449–461 (2012)
Pavón-Pulido, N., López-Riquelme, J.A., Pinuaga-Cascales, J.J., Ferruz-Melero, J., dos Santos, R.M.: Cybi: A smart companion robot for elderly people: Improving teleoperation and telepresence skills by combining cloud computing technologies and fuzzy logic. In: The International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, pp. 198–203 (2015)
Pinikas, N., Panagiotakis, S., Athanasaki, D., Malamos, A.: Extension of the webrtc data channel towards remote collaboration and control. In: The International Symposium on Ambient Intelligence and Embedded Systems (2016)
Pitzer, B., Osentoski, S., Jay, G., Crick, C., Jenkins, O.C.: Pr2 remote lab: an environment for remote development and experimentation.

In: The International Conference on Robotics and Automation (ICRA'12), IEEE, pp. 3200–3205 (2012)
Reed, K.B., Peshkin, M.A.: Physical collaboration of human–human and human–robot teams. IEEE Trans. Haptics **1**(2), 108–120 (2008)
Robot Operating System (2021). http://www.ros.org
Santos-González, I., Rivero-García, A., Molina-Gil, J., Caballero-Gil, P.: Implementation and analysis of real-time streaming protocols. Sensors **17**(4), 846 (2017)
Santoso, P., Khoswanto, H., Sandjaja, I.N.: Web-based robotics laboratory. In: MATEC Web of Conferences, vol. 164, p. 01034 (2018). EDP Sciences
Sayouti, A., Medromi, H., Moutaouakil, F.: Autonomous and intelligent mobile systems based on multi-agent systems. In: Multi-agent systems modeling control, programming, simulations and applications, pp. 451–4682. InTech, London (2011)
Sayouti, A., Medromi, H.: Multi-agents systems for remote control on internet. Int. J. Appl. Inform. Syst. (IJAIS), 36–41 (2012)
Sepulveda, R.R.: Evaluation of teleoperation system performance over a cellular network. PhD thesis, Georgia Institute of Technology (2016)
Simmons, R., Goodwin, R., Koenig, S., O'Sullivan, J., Armstrong, G.: Xavier: An autonomous mobile robot on the web. Beyond Webcams: an introduction to online robots, pp 81–97 (2001)
Stein, M.R.: The pumapaint project. Auton. Robots **15**(3), 255–265 (2003)
Sundaram, A., Gupta, M., Rathod, V., Chandrasekaran, K.: Remote surveillance robot system—a robust framework using Cloud. In: The International Symposium on Nanoelectronic and Information Systems (iNIS2015), IEEE, pp. 213–218 (2015)
Sung, G.T., Gill, I.S.: Robotic laparoscopic surgery: a comparison of the da Vinci and Zeus systems. Urology **58**(6), 893–898 (2001)
Tam, B., Kottege, N., Kusy, B.: Augmented telepresence for remote inspection with legged robots. In: The Australasian Conference on Robotics and Automation (ARAA), IEEE, (2017)
Tan, Q., Denojean-Mairet, M., Wang, H., Zhang, X., Pivot, F.C., Treu, R.: Toward a telepresence robot empowered smart lab. Smart Learn. Environ. **6**(1), 1–19 (2019)
Thangavel, D., Ma, X., Valera, A., Tan, H.-X., Tan, C.K.-Y.: Performance evaluation of MQTT and CoAP via a common middleware. In: The 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), IEEE, pp. 1–6 (2014)
Tiberkak, A., Lemlouma, T., Belkhir, A., Bouabdallah, A., Hentout, A.: A novel approach for generic home emergency management and remote monitoring. Softw. Pract. Exp. **48**(4), 761–774 (2018)
Toris, R., Kammerl, J., Lu, D.V., Lee, J., Jenkins, O.C., Osentoski, S., Wills, M., Chernova, S.: Robot web tools: efficient messaging for cloud robotics. In: The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2015), pp. 4530–4537 (2015)
Tung, A., Wong, J., Mandlekar, A., Martín-Martín, R., Zhu, Y., Fei-Fei, L., Savarese, S.: Learning multi-arm manipulation through collaborative teleoperation. In: The International Conference on Robotics and Automation (ICRA'21), IEEE, pp. 9212–9219 (2021)
webrtc-java (2021). https://github.com/devopvoid/webrtc-java
Witrant, E., Canudas-de-Wit, C., Georges, D., Alamir, M.: Remote stabilization via time-varying communication network delays: application to tcp networks. In: The International Conference on Control Applications, vol. 1, IEEE, pp. 474–479 (2004)
Yang, G., Lv, H., Zhang, Z., Yang, L., Deng, J., You, S., Du, J., Yang, H.: Keep healthcare workers safe: application of teleoperated robot in isolation ward for COVID-19 prevention and control. Chin. J. Mech. Eng. **33**(1), 1–4 (2020)
Yu, R., Huang, X.: Robot remote control internet architecture. In: Information and automation, pp. 514–518. Springer, Berlin, Heidelberg (2011)