SERVICE ROBOTS INTEGRATING SOFTWARE AND REMOTE REPROGRAMMING

Davydov D.V., Eprikov S.R., Kirsanov K.B., Pryanichnikov V.E.

This Publication has to be referred as: Davydov, D[enis]; Eprikov, S[tanislav]; Kirsanov, K[irill] & Pryanichnikov, V[alentin] (2017). Service Robots Integrating Software 4nd Remote Reprogramming, Proceedings of the 28th DAAAM International Symposium, pp.1234-1240, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/28th.daaam.proceedings.172

Abstract

This paper presents a solution to the problems of redundancy and organisation of the software architecture inherent in robotic systems and its' simulators. The analysis and comparison of these solutions with the existing ones are given. It was designed a new software architecture for controlling mobile robotic complexes on the base of two models. Developed simulators and usage of the actor model were the base for creating the technological platforms for distributed software control, providing the simultaneous access of several users to a group of mobile robots and their virtual models.

Keywords: building a network of robotarium; distributed control of mobile robots and their simulation software for mobile robot; reservation methods in robotics; dynamic reprogramming

1. Main problems

When developing intelligent robots, one of the problems is the lack of effective software to create systems of group control. For example, in the well-known software packages (ROS, MRS, and others) does not included specific development tools for control of distributed mechatronic systems, especially in remote mode. They also have a common disadvantage, that is the unreasonably high costs associated with installing, upgrading and maintaining equipment and working in this software environment.

This paper presents the results of the development and testing of the new software-information platform to provide the user with unified access to the software environment to control a variety of robots, including operational and remote changes of executing programs during their functioning.

This enables the following: reducing the cost of setting up executing programs; implementation of operational adjustments during operation of the robots; improving the efficiency of the developed technical tools for teaching and learning processes of the students (see Fig. 1), specialised in robotics and IT- industry, by increasing the use of training equipment due to the paralleling access to it.

Fig. 1. Virtual model of Robocup Logistic competition for educational purposes.

Developed simulators and usage of the actor model were the base for creating the technological platforms for distributed software control, providing the simultaneous access of several users to a group of mobile robots and their virtual models. It's the first time, then under a single software platform are combined programming and control in parallel with a virtual models of mobile robots and their real prototypes "implemented in metal" (see Fig. 2).

Fig. 2. Real robotarium (right side) and their virtual model (left side) with mobile robots "Robotino".

Platform allows you quickly and remotely reprogram systems and significantly reduce requirements to the level of training of specialists-software developers, to ensure the scalability of each component system on physically distributed computers. Thus, significantly decrease the costs of adaptation programs for specific tasks, and radically simplifies development of control systems and user interfaces.

This approach was tested on mobile robots with different kinematic schemes, different types of microprocessor control with the use of sensors, operating on different physical principles. Testing was conducted on robots: AMUR-105, AMUR-107, AMUR-205, three Robotino robots and two small mobile robots, placed in 4 laboratories-robotariums, as well as the simultaneous launch with 80 virtual simulators, operating in real-time. The experiments confirm all these advantages.

Future plans include the expansion of the use of these approaches for teaching and scientific experiments and development of a network of robotariums created by MINOT Russian State University Humanitas, Chair of Sensory and control systems MGTU "STANKIN" (basic department with KIAM Russian Academy of Sciences) jointly with universities in Russia, Croatia (Zadar University) and Austria (Vienna Technical University), including the association with the educational network "Synergy" (project of Concern "Festo"). It is also envisaged to improve further characteristics of embedded in the software platform physical engine, as well as the testing platform in the development of the system production processes with integrated logical control, for example, in automation workshop, implements the concept of "Industry 4.0" in Kimry, realising by International Laboratory "Sensorika".

2. Redundancy processes

The development of mobile robots and robotic complexes is closely connected with the development and implementation of software (SW) specialised and common use SW. In this case SW, focused on mainframes for PC, finds its applications in mobile robots and mobile robotic complexes (RTC).

Unfortunately, most of this modern software can't just be implemented in software without additional modification/processing, so the problem of migration from the world of mainframes in the RTC remains relevant. This paper presents a solution to the problem of redundancy of work processes in mobile robotic complex as well as a new software architecture for geographically-distributed management of a group of mobile robots.

To date, the capacity of modern electronic computing machines (computers) and micro-controllers (MC) allow us to solve a huge number of computational problems. In addition, the size of modern micro-processors and their architecture (ARM, MIPS) allows you to perform a complex calculations in such limited dimensions of the objects as any MRC. Unfortunately, computers and MC are not insured against loss computed results during operation. Reasons for the loss of information can serve as the lifting of external power supply, failure of some hardware, unauthorised system access, error, and other unforeseen situations. To prevent the loss of calculated results are the backup, and periodically saves the data in permanent memory, which can later play to continue after termination of the computation. Such mechanisms include hardware (*RAID arrays, redundant clusters*) and software (*rsync, rdiff-backup*). A brief look at the mechanisms of hardware and software redundancy. *RAID* arrays (redundant array of independent disks, [9]) is a redundant mass of independent disks.

These arrays serves to increase the reliability of data storage and/or to increase the speed of read/write information (or both simultaneously). Today, there are a few of the most frequent types of such arrays:

1. *RAID 1* - mirrored disk array
2. *RAID 2* - arrays using the Hamming code
3. *RAID 3 and 4* - disk arrays with striping and a dedicated parity disk
4. *RAID 5* - disk array with striping and no dedicated parity disks.
5. *RAID 6* - disk array with striping, using two checksums computed by two independent methods;

Failover clustering High-Availability cluster (HA-cluster) is a group of servers, that is designed in accordance with the methods of providing high availability and guaranteeing minimal downtime due to hardware redundancy. Without clustering server failure leads to the fact, that it supports applications or network services are not available to restore its status. Failover clustering corrects this situation, restarting the application on other nodes in the cluster without administrator intervention in case of detecting hardware or software failures. The restart process known as failover. In this process, software clustering may optionally configure the node before starting the application on it.

In addition to hardware redundancy, there are also tools to backup data by program tools. Such tools include many modern software like *Acronics Disk Director, rsync, filezilla*, etc. Such software is responsible for making copy of all processes to self-preserve information from permanent memory. Unfortunately, the processes that are running in memory of a computing device are not responsible for it.

To save the state of the computer, there are hardware solutions for full redundancy. The advantages of hardware redundancy before the software is fully clone all running processes on the remote machine. This requires the availability of accurate copies of the computer, where you stored the entire state of a redundant computer. If you are reserving the N devices, that will require another N exactly the same devices, which entails the problem of excessive use of electricity, large employment areas, which are the device and the cost of the computing complex. When considering computer complex, consisting of several of mobile robots (MR) this option cannot be applied due to its complexity (in front of a personal computer), as when you clone a built-in operating system (OS) the MR will not be taken into account many additional parameters such as sensors readings, the wheels positions because of the inability of the robots at the same time in the same environment and with identical coordinates. In addition, many embedded systems of MR does not have the possibility of hardware redundancy. A possible solution could be the introduction of virtual machines on MR, but this may be difficult due to the large size of the software/hardware and software requirements of MR, which can severely affect the performance of the MR during operation. Software redundancy was also considered in [1], but required a complete restart of the OS that require a long resume.

The new approach of the redundancy, in contrast to the above decisions is that the system redundancy (SR) MR (Master) periodically sends the process state to the neighbouring MR (Slave) directly over a local area network. MR (Slave) monitors the MR (Master) and will either run a copy of the process or will wait for the resumption of MR (Master) from which came the information to send the received information back to resume this process at the MR (Master).

However, the decision about who is the primary (master) object and who is the backup (slave) is able to get out SR or operator. For example, you can assign one of MR major, that if it was not found another similar MR. Thus, when the second MR will be on the network it will go into standby mode. With the simultaneous choice of roles MR (in case of simultaneous receiving the role, each MR) the advantage of choosing the main role will be given to MR, whose name will have less weight W. For example, MR with names $Po6or1$ and $Po6or2$ will have a main and standby role, respectively if the condition $W(Robot1) < W(Po6or2)$.

In this approach, the SR periodically retrieves full information about running process and sends it either to MR (Master), that can play the running programs, or saves on a remote server, that when you resume, MR (Master) will send the accumulated information on this MR (Master). That will allow MR (Master) to continue its work from where you left off. The relevance of this decision is to eliminate the need for hardware redundancy running processes for the OS as well as the possibility of software redundancy running process or multiple processes. This allows you to save the state of processes on the fly without involving a backup device. This approach is also relevant for the redundancy of individual modules in a heterogeneous modular mobile robots [2,3]. However, when transferring the backup information is not desirable the use wireless networks due to the low bandwidth, and also due to less secure wireless networks, which is described in detail in [4] [5].

To be able to reserve a work processes offered two solutions. The first is to add to the process possibility to retrieve the data and send this data to the backup device (MR, RTC server or the other computer), as well as the ability to read data from the backup device to resume operation. The second approach is to extract information on the process (*sections bss, heap, stack*, etc.) by means of the kernel and then sending of location information to MR (Slave). The following is a comparison of these backup options:

Option	Advantages	Disadvantages
Redundant process sends data itself	Not required to retrieve data using OS	The need to change source code of the redundant process. A lack of universality.
Separated the fallback mechanism	There is no need to work with the source code of the program. Versatility.	The complexity of the implementation.

Table 1. Comparing options software redundancy.

The first option does not provide mechanisms to work with the Linux kernel, but it needs an access to the source code of the reserved program. While inside of process needs to be as mechanisms of discharge data and algorithms for reading the data for the possibility of resumption of work. Option becomes impossible if the source code of the program is not available.

The second option: there is redundancy processes, without editing its source code, making this approach universal to any process without providing the source code, but increases the complexity of the implementation of this mechanism with the possibility of running a program in multiple threads, open files, read status registers, cache, etc. In this embodiment, due to the complexity of the implementation, we consider only the redundant single-threaded programs without interacting with the devices input/output and files.

Now fully worked out the first option because of it's relative easy implementation.

3. The architecture of the mobile robotic complex

To refer the software (designed to interface application software with the network and providing unification of interaction between programs in terms of the heterogeneity of the computing platforms) is used the term "middleware software". This type of software acts as subsystems in larger. If in modern database technology (DB) and Internet middleware is studied in detail, as applied to robotics, these studies have not reached the same level. Apparently, it is connected with the fact that until recently, the software for robotic systems was quite simple (from the point of view of software engineering) and does not go beyond organisation of the simplest interactions between the objects of control.

In the process of implementation for system of geographically distributed group control mobile robotic devices [6-8], we analysed the existing systems that provide similar functionality, namely, Microsoft Robotics Developer Studio (MRDS), Robotics Operating System (ROS), and their predecessors, such as Player Project, LAAS GenoM and URBI (see Wikipedia). These systems can be roughly classified into "large", i.e. includes many components for solving various problems: SLAM (Simultaneous Localisation and Mapping), the calibration of sensors, image processing, etc. However, there are "small" robotic frameworks, for example, the Concurrency and Coordination Runtime (included in MRDS), which carries out only the transfer of messages between components. These systems are designed to work within the local area network (LAN) with a stable connection and contain virtually no funds for proper recovery in a data channel. But this problem always arises when it is necessary to provide mobile robotic systems (RTC). Thus, in the process of creating the technology of territorially distributed group management mobile RTC was required to develop a brand new software architecture.

The proposed architecture is based on the principle of organisation of information-measuring and control systems (IMCR) of the robotic complex in the form of a local area network with mobile nodes [4] and the "concept of drivers". Under this concept, every node of such LAN is electronic (e.g., microcontroller) or mechatronic device (e.g., actuators with control system).

Thus a mobile robot (MR) is represented as a composition of several mechatronic devices and their drivers. On the onboard computer [2] mobile robot driver for each mechatronic device is a separate computing process, and collaboration of these processes it is necessary to organise and synchronise.

To solve these problems, we use the following approaches:

- Each command or data from the sensors are time-stamped when passing each node of the LAN;
- Each driver is equipped with its own "Service name", which stores information about the adjacent (from the point of view of the network topology) drivers;
- Each driver is dynamically constructed by the interaction graph, which stores information about the neighbours.
- Uses a text data exchange format.
- Data sharing enables transmission of not only individual teams, but entire programs.

The system of differentiation of access (LDAP — Lightweight Directory Access Protocol) allows you to control the access rights of user groups. Adding time stamps to messages, allows you to accurately synchronise the drivers and take into account the time delays and the difference in time between multiple geographically - distributed nodes in the network. "Name service" is a specialised software that every 20 seconds, scans your network and looks for similar services, and to discover and shares with them all available information in the algorithm is close to distributed hash tables – Distributed Hash Table (DHT). This makes the quality of decentralisation. For example, the robotic system is configured within a specific LAN, no changes can be disconnected from it, and thus will not be disturbed its normal operation. To ensure the integration MRTU in a geographically distributed group, it was necessary not only to implement communication between electromechanical part of the robot and the onboard computer, but also to integrate the IMCR directly into the General management hierarchy.

For a formal description of interaction in multi-agent networks was applied the actor model. As a mechanism of message passing library is used ZeroMQ (<http://zeromq.org/>), which serves as a high-level wrapper over standard network sockets operating system. This mechanism includes a number of functions such as auto-reconnection after breakage with no loss of context, the possibility of tunneling through the lower layer protocols: TCP, IPC, or through PGM (Pragmatic General Multicast). Furthermore, unlike more complex protocols (e.g., Concurrency and Coordination Runtime), ZeroMQ is quite simple, enabling you to deliver it directly into the Arduino microcontroller module c EthernetShield, but with some limitations related to its low productivity. So, when using this microcontroller maximum frequency command in the test implementation was 20 Hz.

In figure 3 is shown the general scheme of interaction: two mechatronic devices rc1 and rc2 (RobotComponents) asynchronously pass the procedure of registration in the name Service ns (NameServer). Then the control system (CS) is requesting a list of registered devices, and performs synchronous and asynchronous communication with the mechatronic devices. Filename in this case is the IP address and port, arbitrary text, e.g. "engine", or a path in the hierarchy, e.g., "University→Laboratory→Robotino→Engines→3→PWM".

Fig. 3. The mechanism of network interaction in the framework of the proposed architecture.

Driver for each device provides:

- naming and identification (method of obtaining access to the device by name);
- transmit and receive commands, including the other drivers.

Commands and data are transmitted in JSON format (see Fig.4), which is quite simple and can be implemented on low-power processors, including AVR (Arduino). JSON is a text (ASCII) format for the presentation of structured data, which allows you to group a list (vector, array) and associative arrays, i.e. sets of pairs “key-value”, which gives the possibility manually generate control commands, without resorting to specialised libraries, as required in other similar systems. Also, there is the opportunity to annotate transmitted data with additional information without modifying the host software, or to arbitrarily change the order of fields within a single message. This makes it possible to organise the transfer of not only individual teams, but entire programs, that immediately on receipt executed by the interpreter, operating on the onboard computer [2]. Thus arranged dynamic re-programming of mechatronic devices, i.e. without stopping or “reloading” on-Board computer of the MR device, you can update it or correct it's work. This greatly simplifies the simultaneous support of different versions of the same driver in a network of mechatronic devices in the task of geographically distributed control. The redundancy of information, that occurs due to its text representation, has no significant effect on throughput, since teams usually compact even in the textual representation and are in comparable size to the header of the IP packet (20 bytes).

Fig. 4. Diagram - transmission of messages: device generates a messages, which are encoded in JSON and are passing to ZeroMQ.

In that case, when the amount of JSON-encoded data begins to have a significant impact on processing speed on the data transfer, we can use format is binary AVRO [8], however its implementation on the AVR is not yet. For visualisation of discovered network components, display operator interface control software was created. It represents all the detected components as a tree view, as shown in Fig.5.

Fig. 5. One of the windows on-screen interface presents (in a tree view) detected robot AMUR-7 components (located in lab1 RSUH): ultrasonic sensors (US), electrical motors (Engine), TV-camera (Camera) and rechargeable battery (Battery).

4. Conclusion

In the course of work was drawn up two models of software redundancy process: as to modifying the source code of reserved process and without it. This technique allows to get rid of the need for hardware redundancy and to increase the ability of software to back up data on a computer and MP running on the Linux OS. At the moment a check process without editing its source code can be applied only to processes with a single flow and is not working with file descriptors or devices because of the complexity of the redundancy module, therefore, the support of multithreaded programs and programs, that work with other OS resources such as files, devices, I/O will be developed in the future.

The proposed software architecture has proven its effectiveness in the creation of geographically distributed control systems academic group of mobile robots, consisting of 2 robots of the type "AMUR" (the Autonomous Mobile Universal Robot, International Laboratory "Sensorika") and 3 robots "Robotino" (the German company Festo). The system operates between MSTU "STANKIN" (Moscow) and the International Institute of new educational technologies (IINET) Russian state University for Humanities (Moscow). The same channel also operates between Keldysh IAM RAS (Moscow) and the Far-Eastern Federal University (Vladivostok) [10-14]. The information and communication architecture can be used in the system of distance education [4,10] to organise the access of students to real mechatronic systems of different models and manufacturers.

5. Acknowledgments

The work was carried out with the partial financial support of the RFBR, projects №16-07-01264, 16-07-00935, 16-07-00811.

6. References

- [1] V.Ya. Hecha, S.G. Sementsov, S.A. Kozlov, A.M. Konoplev. The security Concept for Linux-like operating systems in control systems and protection of nuclear reactors // Problems of electromechanics. Proceedings VNIIEМ, 2004.
- [2] (В.Я. Геча, С.Г. Семенов, С.А. Козлов, А.М. Коноплев. Концепция обеспечения безопасности Linux-подобных операционных систем в системах управления и защиты реакторов АЭС // Вопросы электромеханики. Труды ВНИИЭМ, 2004).
- [3] Kirill Kirsanov. The Architecture of Robotics Control Software for Heterogeneous Mobile Robots Network / Kirsanov Kirill // Procedia Engineering 01/2014; 69:216-221. DOI: 10.1016/j.proeng.2014.02.224.
- [4] Andreev V. P. Group control of mobile robots by means of Ethernet technology / P. Andreev V., Pryanichnikov V. E., Prysev E. A. // Proceedings of XXI International scientific-technical conference Extreme robotics. – SPb.: Publishing house "Politehnika-SERVIS", 2010. – P. 427 – 430.
- [5] Education on the basis of virtual learning robotics laboratory and group-controlled robots / Victor Andreev, Sergey Kuvshinov, Valentin Pryanichnikov, Yury Poduraev // 24th DAAAM International Symposium, 2013. Procedia Engineering, 2014. – V. 69. – P. 35 – 40.
- [6] Andreev, V.P., Kirsanov K.B Technology Supervisory control of mobile robots through Internet// Proceedings Int.conf. "Extreme robotics-2015". -SPb, "Politehnika-Servis", 2015.-P.270-272.
- [7] Wolfgang Emmerich, The impact of research on the development of middleware technology / Wolfgang Emmerich, Mikio Aoyama, Joe Sventek // ACM Transactions on Software Engineering and Methodology. – N. Y.: ACM, 2008. – T. 17. – No 4. – P.19 – 48.
- [8] Zareshin S.V., Shustova L.I., Shestakova N.Y. Comprehensive analysis of wireless networks security in moscow central district //Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian. – IEEE, 2017. – P. 240-241.
- [9] Anastasia A.V. et al. Analysis of security of public access to Wi-Fi networks on Moscow streets //Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE Conference, Russian. – IEEE, 2017. – C. 105-110.
- [10] Description of RAID arrays - <https://ru.wikipedia.org/wiki/RAID>, 01.11.2017.
- [11] V.E. Pryanichnikov, B. Katalinic, A.A. Kirilchenko, R.V. Khelemendik, S.V. Kuvshinov, D. Vician, A. Uglesic. New Creative Educational Technologies for Inter-university Network. Elsevier. // Procedia Engineering Volume 100, 2015, Pages 259–268. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. URL: <http://www.sciencedirect.com/science/article/pii/S1877705815003938>
- [12] Kirillchenko A.A., Pryanichnikov V.E., Rogozin K.V. Limits of validity and reliability of proofs. Scepticism in mathematics, functions, and traditions // Information-Measuring and Control Systems. 2013. V. 11. № 4. P. 57–65.
- [13] Frolov A.A., Biryukova E.V., Bobrov P.D., Mokienko O.A., Platonov A.K., Pryanichnikov V.E., Chernikova L.A. Principles of neurorehabilitation based on the brain–computer interface and biologically adequate control of the exoskeleton // Human Physiology. 2013. V. 39. № 2. P. 196–208.
- [14] Katalinic B., Pryanichnikov V., Ueda K., Cesarec P., Kettler R., et al. Bionic Assembly System: hybrid control structure, working scenario and scheduling // Proceedings of 9th National Congress on Theoretical and Applied Mechanics. Brussels, 2012. P. 101–108.
- [15] Vladimir F. Filaretov, Valentin E. Pryanichnikov. Autonomous mobile university robots AMUR: Technology and applications to extreme robotics. // Elsevier. Procedia Engineering Volume 100, 2015, Pages 269–277. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. URL: <http://www.sciencedirect.com/science/article/pii/S187770581500394X>