Proceedings of the FSR'99, Robotics Applications for the Next Millenium, Pittsburgh, PE, pp. 168 - 173, 29.8. - 31.8.1999

ROBODIS: A dispatching system for multiple autonomous service robots

Hartmut Surmann
GMD - German National Research Center for Information Technology, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany, hartmut.surmann@gmd.de

Markus Theißinger
Codit - Informationstechnik GmbH, Ahrstraße 1, D-53757 Sankt Augustin, Germany, markus.theissinger@gmx.de

Abstract

A web-based dispatching system for autonomous mobile service robots is presented. The system is composed of an extensible number of physical and software components. They represent and control objects like robots, doors, cameras and elevators or software components like a job scheduler and a database manager. Components communicate via radio link and Ethernet. Autonomously operating components and additional security features result in an adaptable, highly available, multi user/multi service system. In contrast to other web-based systems the mobile robots act really autonomously and are not teleoperated by users.

1 Introduction

RoboDis is the first web-based dispatching system for multiple autonomous service robots executing transport jobs. Since J. Engelberger 1993 presented the first commercial service robot HelpMate [**?**], such robots gained increasing interest. Meanwhile there is a number of different robot platforms available. The transport jobs for these platforms are either entered directly at the robot via a small keyboard or touch screen or they are firmly pre-programmed. With such an approach a flexible and universal access to the robots is not possible.

Some systems offer a web interface to the user like [**?**], to teleoperate the robots. While this is a big improvement in the user interface, these systems were not designed to extensibly integrate a large number of really autonomous robots which are not teleoperated. To construct a flexible and global transport service with robots, the robots have to be integrated as a module into an entire management system for buildings.

The management system for buildings is composed of several different components. These are corridors, fire doors, elevators, robots and observation cameras, i.e. all sensors and actuators that are available in the building as well as the different users. Both, sensors and actuators and all users are connected via intranet/internet. Parts of the network are realized as radio links, e. g. radio Ethernet. A web interface allow users to enter transport jobs which are stored in a database.

The main characteristics of RoboDis are:

- Autonomously operating components: components decide on their own, i.e. the control logic for decision making is implemented locally on the component.
- Decentral system organisation: functionality and control is distributed among components.
- Web-based user interface: service requests can be entered at any place and time where web access is offered.

A centralized structure of non-autonomous units is inadequate for an adaptable system because the complexity of a building with its many different items leads to a complexity of the total system that is not desirable. A central control component would easily become a bottle neck for the information flow in the whole system. It would also be difficult to extend such a system by plugging-in new components. Therefore the autonomous approach has been selected.

The dispatching system for multiple autonomous service robots introduced here is client-server based. Transport jobs for the service robots can be entered by means of a standard WWW client such as Netscape Navigator or Internet Explorer. In our system more pretentious requirements are made for the term service, i.e. the total system must be highly available, user friendly, safe and transparent. Transparency means, a simple view of the system is given to the user: a specific list of transport jobs is defined by the user and executed in time. System complexity, e.g. specifying which robot and how a robot executes a job, is hidden from the user. [1]

In order to implement such a universal dispatching system, two problems have to be solved. The vehicle routing problem, i.e. the different transport jobs should be optimally distributed among the different robots in real-time and the reliability of the system. Among reliability the following service and safety aspects are summarized:

- High availability of the system: The system should be at disposal 24 hours a day to all users.

[1] The dispatching system can be tested under URL http://lamu.-gmd.de:8080/. Unregistered users may enter transport jobs using the account guest with password guest. However these are not executed by the robots. Since orginal server qingdao was stolen on 10.06.1999 some services are restricted. Since all loging informations are lost too, the only thing we can say is that the uptime of the web server was 5 month with one short reboot to integrated new hardware.

- Authentification of the users: The user who charges the system with a transport job, must be uniquely identifiable. 180 computers in our offices are connected to the system.
- Privacy: Encryption of the transport jobs. Users may access only the jobs charged by themselves.
- Integrity of the transport jobs: The web server has to receive the jobs exactly as they were sent by the user.
- System administration: a large system with numerous robots must support treatment of unforeseen situations and overcoming system errors.

This paper introduces the system and describes the aspects of reliability. The vehicle routing problem will be treated in another paper. Section 2 gives an overview of the physical components of the system. Section 3 presents aspects of the implementation to achieve the above goals. Section 4 summarizes the advances of the system.

2 System elements

Fig. 1 illustrates physical and software components of the system. The mission server fetches unfinished jobs and schedules them for execution by some robot depending on the availability, the position of the robot and on the execution time window of the job defined by the user. The radio server connects the other system components through radio links, e.g. it transmits jobs to the robots, informs the mission server about robot positions and notifies the door and elevator server about the state of each door and elevator respectively. The door server stores the state of each door and decides when to change it, i.e. when to open/close doors. The elevator server similarly operates for the elevators, e.g. when a robot requests an elevator, the server decides whether this is in conflict with human transportation requirements of the elevator and eventually serves the robot.

While each robot forms a control unit of its own, other control units like the elevator server support more than one physical component. This reduces complexity by reducing the number of system components. In a system with a larger number of physical components, several server instances can run parallelly, e.g. one elevator server for each building. A cluster structure can be used for the parallel servers to keep communication requirements low.

2.1 Area of operation

The ARIADNE robot dispatching system consists of independently operating indoor service robots. Each robot moves along plain floors, carrying items or executing observation orders. An example operational area for the dispatching system is a customer service system in a hotel. Customers enter delivery orders through in-room PC/TV sets

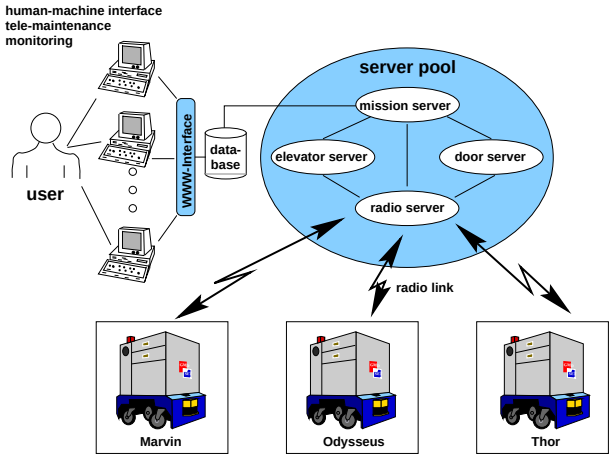

Figure 1: System structure overview

while the service robots collect the desired items. Another example is a patient care system in a hospital where robots deliver documents to physicians, medicine and food to patients or guide patients from one station to another. Other areas are mail delivery, plant nursery, building surveillance, nuclear power plants etc.

Of course, some of these systems could also be built with fixed transportation tracks but a system of autonomous robots is much more flexible an can be adapted to different buildings, cargo sizes and other changes in the environment.

2.2 Service robot team - ARIADNE

Each ARIADNE service robot is capable of unguided collision free navigation and exploration of unknown building structures at a single level. In known areas, the robots can use doors and elevators by radio control and move at several levels. The robot determines a door position from a map, approaches the door and sends an open-door command. When the robot detects the opening by its laser sensor, it passes the door.

Besides navigating autonomously [**?**] the robots can also navigate along a marked optical line. The optical line is mainly used for high precision maneuvers like docking at the automatic battery charger or passing through a very narrow door. The system can switch automatically from autonomous to guided navigation.

The ARIADNE team consists of three mobile robots (Fig. 2): Odysseus, Marvin, and Thor. Each is about 80 cm $\times$ 60 cm large and 90 cm high. The mobile platform can carry a payload of 200 kg at speeds of up to 0.8 m/s (about half the speed of a pedestrian). The right and left driving wheels are mounted on a suspension on the center line of the mobile platform. Passive castors on each corner of the chassis ensure stability.

The core of every robot is a Pentium PC 166 MHz with

Figure 2: The ARIADNE-team: Odysseus, Marvin, and Thor executing service orders. Each robot is equipped with four drawers, two on each side

Figure 3: Battery refill station

16 MB RAM running under real-time Linux. Two microcontrollers are used. One controls the internal states, display, keyboard, and radio link of the robot. The other one manages the motors and the optical line-tracking. Each platform is rigged with two laser scanners, one on the front and the other on the rear of the robot. Each laser scans 180° of the environment in front of it every 80 ms. The processing time for a complete scan of 361 distance values is 170 ms yielding 6 scans per second [**?**].

Each of the 250 kg robots can operate for about 8 hours on one battery charge. When the power drains, the robot visits a power station, connects itself to it and recharges his batteries (Fig. 3). The mission server will be informed about the time period for which the robot is unavailable and another robot takes over.

2.3 The building

The robots currently move in the GMD-ROBOBENCH, an H-shaped building of about 800 m^2. The building is departed into 84 office rooms and 15 corridors on 3 levels reachable through elevators. All elevators and 17 doors can be automatically controlled by every robot. Office room doors can be opened by humans only. Web cameras on some floors allow live observation of robot actions.

A floor plan of the building is shown in the internet. It was copied from the construction plans of the building. During execution, the robots need to have a map of the building together with some additional database information e.g. telephone lists which is linked to the map. The map is represented as a topological map in a graph like structure with additional geometric informations. During initialization, the robot can construct a floor plan by travelling through each floor and tracking coordinates. The link to the telephone list and the codes for opening doors and selecting elevators have to be given by an expert. In the future, a camera-equipped robot will read the sign on a door using OCR when reaching an office. This enables the robots to build a database of offices with coordinates and names of the persons working in the offices by themselves.

3 Service and reliability aspects

Each system unit operates autonomously. There is no central component of control. Each unit collects the necessary information to do its job on its own. For example, a robot requests an order from the vehicle routing program (Fig. 4). The scheduler considers the robot position and the list of transport jobs and sends the most appropriate job to the robot. The robot is responsible for all details of job execution and at the end notifies the scheduler that the job is either completed or failed.

3.1 High availability

A user enters transport jobs through a web interface (Fig. 5). Among other things, redundancy provides high availability. Several PCs can be used as a hardware platform for the web server. Here, standard PCs with the Linux operating system (v. 2.0.35) and Apache web server (v. 1.3.4) have been selected.

Linux provides a fast and reliable disk system by supporting software RAID [**?**]. RAID level 5 allows the system to continue an operation even if one of its hard disk fails. The database ensures the integrity of the transport jobs with its atomic transactions: each job is either stored completely or not at all. Even in case of a failure there is no in-between state with corrupted data.

The Apache server as well as the stateless HTTP-protocol increase the availability because inquiries are forked to independent child processes. If one of these child processes dies,

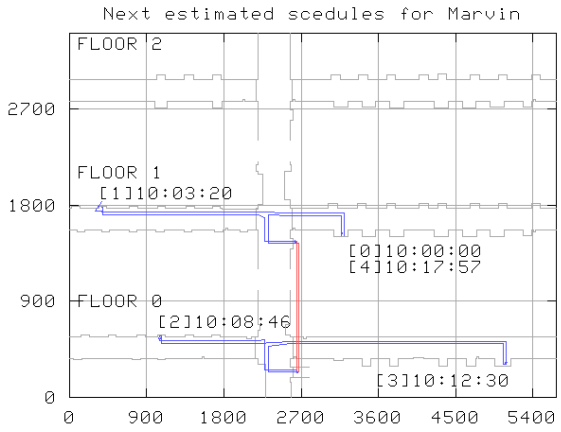

Figure 4: List of scheduled jobs

Figure 5: Web formular for transport jobs

the total system is not endangered. A further aspect for high availability is the information throughput, which amounts up to 1000 inquiries per minute for the current implementation. The Apache server is world-wide the most frequently used web server. In particular the source code is available and therefore its bugs are well-known and documented.

Software components of the building management system are implemented as CGI scripts [**?**], [**?**]. This additionally increases the service. So on the one hand, data (jobs, robot positions, states of doors and elevators etc.) can be stored in and queried from a database. Detailed debugging protocols can be created from the database if necessary. On the other hand a modular system is achieved, which can be extended de-centrally. Further components are easily added by implementing new CGI scripts (add-ons).

3.2 Authentification of the user

Username and password are required for authentification to the system. A user, who places a transport job, must be identifiable definitely by the system. Information about the state of the transport job is accessible only for the system administrator and for the user who placed the job. The user is responsible for the correctness of the job.

The web server requests username and password from the client's browser. The name is stored in the database in conjunction with the job and is checked for every further access to the database.

Passwords are stored encrypted in the usual password files. Passwords have to be selected carefully, since they are crucial for security. Therefore, when the administrator creates a passwords or a user changes it, the system prescribes a minimum length and forbids words from a dictionary. Furthermore the system administrator is informed automatically, if more than five attempts to authenticate fail. For high security requirements it is possible to allow only to certain IP addresses to access the system.

If the web interface is to be used in an intranet only, the encryption system described below can be configured for two sided identification, i.e. each client uses its own certificate and all client certificates are known by the web server. In this scheme, the effort of entering username and password can also be avoided, if each certificate is used to uniquely identify a user.

3.3 Privacy

Privacy is achieved by encryption. Since a web-based client-server system is realized, the Secure Socket Layer protocol (SSL) [**?**],[**?**] is used for the encryption. All data that requires an authentification is transmitted encrypted by the client to the server. So transport jobs, order lists (Fig. 6), deletions of jobs, passwords, password modifications etc. are protected against reading by unauthorized persons. The SSL protocol is particularly user friendly, since the encryption is done by the browser and web server and is transparent for the user. The software in use is SSLeay (v. 0.9b) with the appropriate Apache patch. The server uses self-created certificates. On server side encryption up to 128 bits is possible, if the browser supports this. Otherwise the keys have a minimum length of 40 bits.

3.4 Integrity of data

Integrity of data must be ensured during the transport from the user's browser to the server. I.e., the data must be received by the server exactly like it was sent by the browser. Manipulations and loss of data during the transfer must be excluded or detected.

Transport jobs from the user to the server are sent with

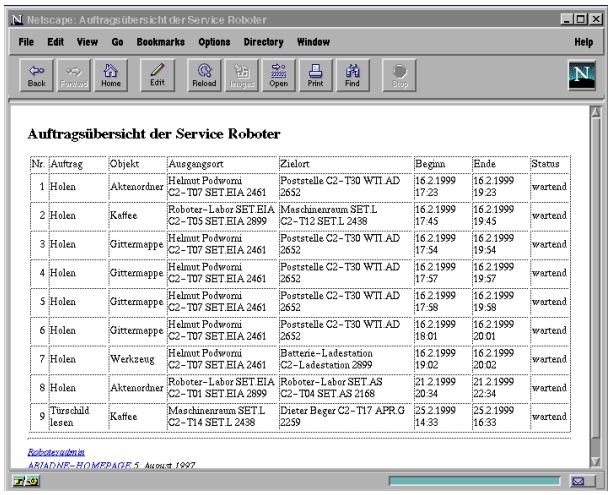

Figure 6: Job lists

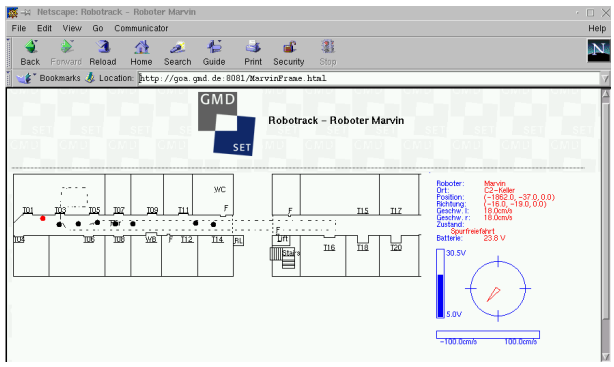

Figure 7: 2D-robot position tracker

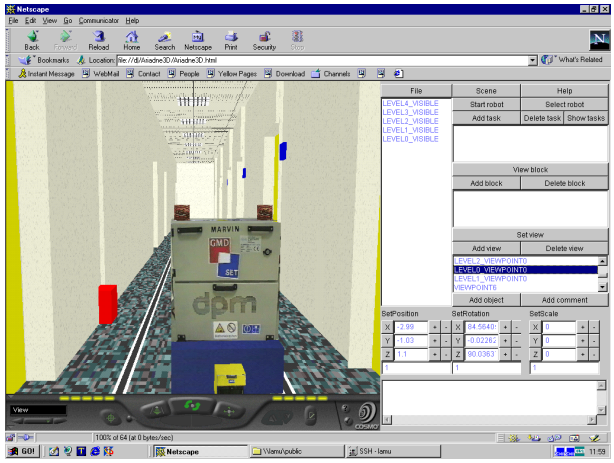

Figure 8: 3D view of the robot while driving through the building.

the POST method as parameter strings. The server checks the input length of the parameter string. The user's input is syntactically checked on application level by the CGI scripts. The syntactic correctness of date and time as well as the start and destination addresses are verified. They are compared to telephone lists and room plans of the building. A fuzzy matcher facilitates the selection of correct parameters for the user. Manipulated data with a high probability of syntactical incorrectness is rejected on the application level.

Both the parameters and the length of the parameters are encrypted during transmission. This principally offers protection from unnoticed falsification of data. One receives only absolute protection from integrity loss if hash functions and digital signatures are implemented. Both will be integrated into the system in future if necessary.

3.5 System administration

Usually the whole systems operates unattended. However, the system offers special control options for an administrator so that human interaction is possible in emergency situations. The are two different types of emergency situations:

- Emergencies detected by system self tests.

  Each robot sends a heartbeat, i.e. a position change information while it is moving and some other signal if it is standing, e.g. while charging its battery. If the heartbeat stops and the robot is unreachable, the administrator will be alerted immediately. If the unavailable robot currently was executing some task, the administrator can schedule this task for another robot. An additional task might be necessary for finding the defective robot and unloading what it was carrying.

- Emergencies detected by human administrators.

  Java applets and a VRML browser with EAI interface (e.g. cosmo player) visualize the behavior of the robots two- and three-dimensionally and increase thereby the user friendliness of the system (Fig. 7, 8). An administrator can check these and the live camera pictures to detect, whether a robot reaches a critical area, e.g. an area crowded by persons or an area of construction work.

  The administrator can switch to guided mode and issue navigation commands (Fig 9). The commands will be send by radio to the robot. If the robot does not react, a person has to walk to the robot and perform actions, e.g. reboot the robot operating system. Mechanical failures of the robot movement system are rare and repaired by the manufacturer of the robot.

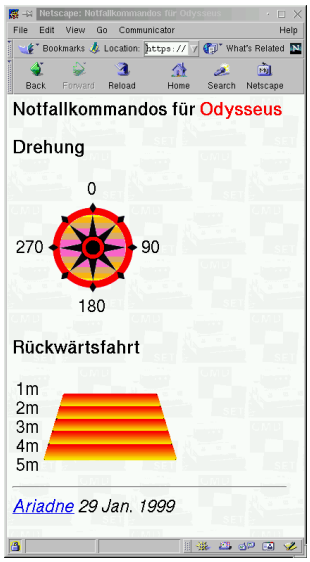

Figure 9: Emergency navigation control

4 Conclusion

A management system (ROBODIS) for the dispatching of multiple service robots is a demanding development goal. To simplify system design all components of the system operate autonomously (not teleoperated) and communicate through Ethernet. Modern technologies like HTML, Java, SSL ease the construction of a sophisticated user interface which enables company-wide use of the robot dispatching system.

Additional robots can be easily plugged into the system due to the soundness of the autonomous approach. As future work it would be desirable to automatize the system extension, i.e. if the system detects the heartbeat of a new robot it will automatically supply new jobs to it, create new HTML observation pages and so on. Also, outdoor driving of the robot is under construction.

Acknowledgments

Special thanks to Claudia Hirsch for hints on security and paper corrections.

References

[1] Joseph F. Engelberger, “Health-care robotics goes commercial: the ‘HelpMate’ experience”, *Robotica*, vol. 11, pp. 517–523, March 1993.

[2] Ken Taylor and Barney Dalton, “Issues in internet telerobotics”, in *FSR’97 International Conference on Field and Service Robots*, dec 1997.

[3] Hartmut Surmann, Jörg Huser, and Liliane Peters, “A fuzzy system for indoor mobile robot navigation”, in *Fourth IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 95, Yokohama, Japan*, 1995, pp. 83–88, arwarded with *Robot Intelligence Award*.

[4] Michael Pauly, Hartmut Surmann, Marion Finke, and Nang Liang, “Real-time object detection for autonomous robots”, in *Informatik Aktuell. Autonome Mobile Systeme, 14. Fachgesprch.* 10.1998, pp. 57–64, Springer Verlag.

[5] Linas Vepstas, “Raid solutions for linux”, 1999.

[6] *HTML and CGI unleashed*, SAMS net. pub., sep 1996.

[7] Open Market Inc., “The unofficial fastcgi homepage”, 1998.

[8] Netscape Communication Corp., “Ssl version 3.0”, 1998.

[9] Ben Laurie, “Apache-ssl”, 1998.