Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

A Scalable Approach to Human-Robot Interaction

Ashley D. Tews, Maja J. Matarić and Gaurav S. Sukhatme

USC Robotics Research Lab, Center for Robotic and Embedded Systems,
Computer Science Department, University of Southern California,
Los Angeles, USA
{atews| mataric| gaurav}@robotics.usc.edu

Abstract

Much of the current research in human-robot interaction is concerned with single systems and single or few users. These systems and their interfaces are generally tightly-coupled and well-defined. For large-scale human-robot applications, the systems may be unknown prior to designing the interface for potential human interaction. This presents a difficult goal for allowing multiple users to interact with many possibly unknown systems. In this paper, we present an interaction infrastructure aligned with providing this interface. It operates in two phases that accommodate both many-to-many interaction and generalized, one-to-one interaction between users and robotic systems. Our previous research has demonstrated the infrastructure to scale to a large number of users and several systems in simulation. The experiments in this paper substantiate these results in a smaller-scale real robotic environment.

1 Introduction

Typically, research into human-system and human-robot interaction relies on the constraint that the interaction requirements of the systems are known in advance of designing the interaction mechanism. When considering large-scale human-robot applications (on the order of hundreds of entities), this assumption may not be valid since there can be many different types of systems with different interaction requirements. Therefore, the interaction mechanism has to either cater for every potential type of interaction that is expected to be encountered, or provide a more general mechanism that allows basic interaction with potentially any system.

We define multi-user, multi-robot interaction as *the ability for multiple users to interact with and control multiple robots simultaneously*. This definition requires addressing issues about the types of interaction, anticipated numbers of interacting entities, and the heterogeneity of systems. From analyzing these issues, we have determined that an interaction mechanism must be simple for scalability purposes but provide adequate functionality to contend with the possible types of interaction and systems. Our primary assumption for dealing with these diverse issues is that the systems are relatively autonomous. We also assert that an interaction mechanism should provide extended functionality to take advantage of the opportunities of collaborating with multiple systems.

We have designed an interaction infrastructure with these goals in mind. The infrastructure consists of a two-phased approach that allows the exposure of multiple robot systems to multiple users in the first phase, and individual user interaction with systems in the second. It also provides the extended functionality of task queuing and linking. Our previous work demonstrates the infrastructure's operation for tens of simultaneous users and services offered by robotic and embedded systems, and the advantages of queuing tasks [16]. The infrastructure's task-linking ability is demonstrated in [17]. The research presented in this paper substantiates these experiments in the real robot domain with a smaller number of robots and users. Both scalability and task-linking experiments are reproduced and demonstrate the infrastructure as being a practical, scalable interaction mechanism.

2 Related Work

The discussion of scalability in human-robot interaction can be broadened to include other systems since, in a large-scale application, not all of the entities will necessarily be humans or robots. A system can be considered to consist of either physical (such as robots or embedded devices) or virtual entities (such as the services provided by computer programs). As such, human-system interaction covers several large areas of research. In [1], a taxonomy is presented categorizing human and intelligent system research. *An intelligent system* is considered to be one that allows user interaction (e.g., tele-robotics [11]).

In this section, we classify human-system interaction architectures in terms of their scalability and assume that a system consists of a set of nodes (e.g., robots) that provide a set of services to users.

Single-user System Architectures

An important issue that affects the scalability of an approach is how *tightly-coupled* the users are with the system during interaction. This concerns both access to, and control of, system resources. Single-user systems can allow the user full access to their resources. Adding more users means these resources have to be shared, which may be detrimental or prohibitive for some types of interaction, particularly those that require a high and dedicated communication bandwidth.

Research into multi-robot control provides examples where robots can be controlled individually or as a group by a single user [2, 3, 8, 14]. In each of these systems, the robots have a certain degree of autonomy and the user has either high-level group control, or low-level control of individual robots. This indicates that the larger the number of entities that an interaction mechanism has to deal with, the looser their coupling should be.

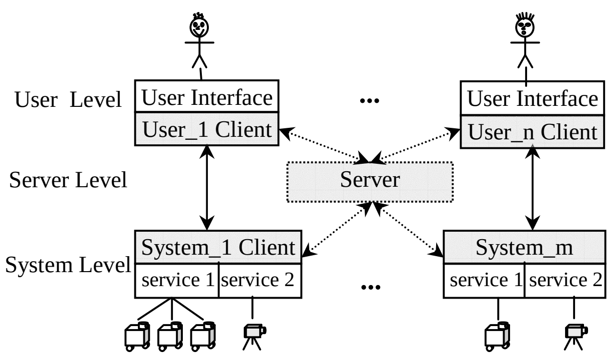

Figure 1. An overview of the infrastructure for connecting multiple users to multiple system services. Many-to-many interactions are shown by broken lines and potential one-to-one interactions by solid lines.

Multi-user System Architectures

Multi-user and/or multi-system interaction is forced to address issues of scalability, resource allocation, and distributed control. A key concern is how to manage the information and communication pathways for multiple entities. Common approaches involve using multi-agent architectures and the Internet.

Multi-agent architectures such as JINI, CHI [7, 12] and Icrafter [13] offer services to users with suitable access to the systems they serve. An example of their application is in interactive workspaces [5, 10] which consist of embedded devices in dedicated environments that provide services to users. The users typically must physically be present in the environment to access the services or have specific software overheads implemented in order to use the interface mechanism.

Small scale multi-user, multi-robot architectures are demonstrated by [4, 15]. These systems use the Internet as a communication medium for commanding robots from remote locations.

Many of the examples presented in this section are application- or domain-dependent. If they allow multiple users, it is typically in pre-determined, small numbers. The interaction infrastructure we present in this paper relaxes the constraints of the number of users and systems and reduces the software overhead required to allow interaction.

3 Infrastructure Overview

By analyzing existing research into multi-user, multi-system architectures, it is realized that many, such as JINI, require the systems to conform to the architecture. The overhead involved can reduce the scalability of a system. By assuming that many systems involved in large-scale interactions are autonomous, a simpler and less constrained approach can be taken. It is with this rationalization that our infrastructure is based on a server/client architecture, as shown in Figure 1.

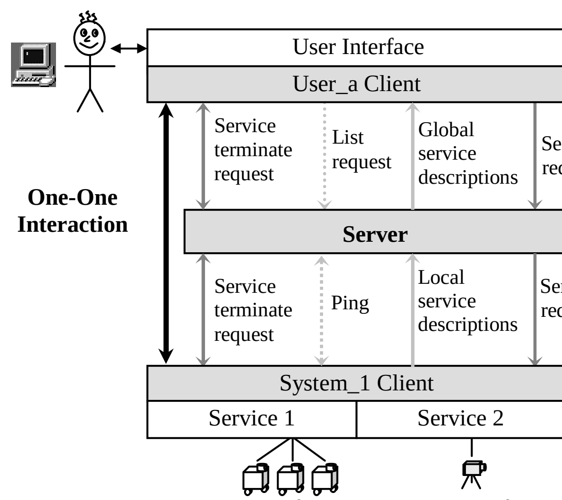

Figure 2. Signal and data flow through the infrastructure between a single system and user. The dashed lines are signals and the solid lines are messages.

A system is typically a robot, robot group, or embedded system (e.g., a security camera network). Each entity (system or user) has a client that interfaces with the server and manages low-level signal and message passing as shown in Figure 2. It is assumed that the server's contact information is known to all systems and users wishing to connect to the infrastructure. The server is the centralized hub of operations that manages the allocation of systems' services to users’ requests. The data flow throughout the infrastructure is by UDP communication.

There are two phases to the infrastructure's operation: many-to-many and one-to-one. The first phase establishes indirect contact between users and systems via the server. Each system upon connection provides a list of its offered capabilities – termed 'services' to the server and whether a task queue should be set up for it. The service list contains the description and type of each service with types being either 'data' or 'command'. Data services are typically provided by autonomous or stand-alone system nodes and inform the user of their task status during operation. Command services require the user to have a more active role by providing information pertaining to their tasks and are typically semi-autonomous.

The server maintains a database of all systems, their service lists, and contact information. Systems are routinely contacted to ensure they are still serviceable. Any that do not respond in a pre-determined amount of time are assumed to be delinquent, removed from the server and notified. They reconnect if they are found to still be operational. This disconnection/reconnection does not affect existing one-to-one interactions between systems and users.

When users connect to the infrastructure, they are provided with the current global list of all services in the server's database. The list is presented to them as a menu of options and includes the extended options of refreshing the list, service-linking, and exiting; the former two are discussed next.

Users need to periodically refresh their service lists since systems and services are added and removed constantly from the server's database. Hence, the users only have a 'snapshot' of the database at any point in time.

Linking services allows the user to combine two services to perform a higher-level function. The services consist of a command and data type. The data service provides relevant information to the command service. They must be semantically compatible with each other to allow service linking. This is currently the user's responsibility.

The second phase of operation occurs when the user selects a service from their menu. The selection is processed by the server to determine whether the service is available or not. The service is available if the user is the only one on an established task queue, or there is no queue. If the service has been removed from the server or is unavailable, the user is notified. Otherwise, the request is forwarded to the applicable system. A system- and service-dependent level (coupling) of interaction is then established directly between the user and the system. The communication bandwidth between interacting entities may be reduced if the user selects multiple services, or the system is managing multiple requests. Once a service has been selected, the system may send a signal to the server to remove it from the database. This mainly applies to systems and services that can only manage a single request. Interaction continues between the service and user until either terminates it. The user may select services at any time, resulting in one-to-many interaction. More details on the infrastructure's operation are provided in [16].

4 Experiments

In previous experiments, we have demonstrated the infrastructure's scalability [16] and service-linking ability [17]. The scalability experiment demonstrated over 900 users using the infrastructure to access 80 services offered by robotic and embedded systems. The service-linking experiment demonstrated the advantages of combining services through the infrastructure. This involved a robot tracking system's service providing target location information to a robot offering a delivery service. These experiments were carried out using the Stage simulator and Player devices [6]. Stage has been demonstrated on many projects to provide a valid representation of the real operation of Pioneer robots. Player provides a suite of software clients that represent the handlers for the devices on the robots such as sonar, SICK laser range-finders and color cameras. The combination of Player and Stage allows programs developed and tested in Stage to transfer readily to the real robots.

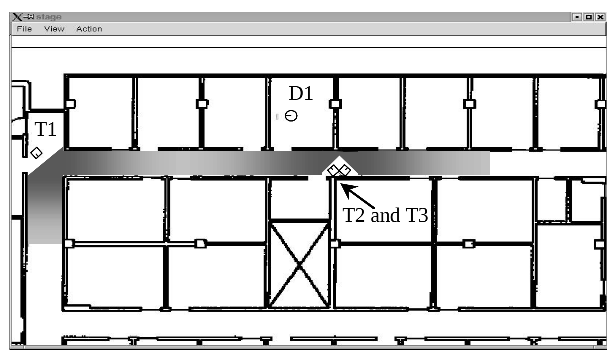

Figure 3. The Stage simulated view of the office floor environment showing the tracking robots (T1-3) and delivery robot (D1) in their initial positions. The gray areas indicate the sensor ranges of the tracking robots.

The experiments in this section substantiate the previous simulation results on a smaller scale in a real environment. Figure 3 shows the environment with the robots and the tracking system's coverage. Four Pioneer 2 DX robots are used to provide tracking and delivery systems. Each robot is equipped with an on-board sonar ring, SICK laser range-finder and Sony PTZ camera. The tracking robots use the cameras and ACTS vision system to locate and identify colored beacons on mobile targets. The laser range-finders are used with the robot's odometry for laser beacon-based localization and navigation. The sonar is used for obstacle avoidance.

The first system is a multi-robot, multi-target tracking system [9], configured to detect three different targets identified by colored beacons. Hence, three services are generated and provide target location information. For the experiments, the system consists of three robots configured to be either stationary or dynamic trackers. Using the infrastructure, the system client can manage up to 10 requests simultaneously.

The second system offers one service and consists of a delivery robot that navigates to goal locations provided by the requesting user. It provides task status details of proximity to the goal location and the robot's current pose. A new goal location can be entered at any time. The delivery robot has a 'home' location in an office where it returns to whenever it is not tasked. For long duration experiments, this location could contain a docking station for recharging.

The two systems share the same environment map and coordinate reference frame so location information is common between them. This map was used in the earlier simulation experiments. No calibration had to be carried out for real robot localization, which is a testament to the accuracy of the map used in Stage and the effectiveness of using Stage for real robot system development.

Experiment 1 – Small Scalability Analysis

The first experiment involves several users requesting tasks from the systems. The three tracking robots are configured to act as stationary sensors. The area of the tracking system's coverage is shown by the gray areas in Figure 3. The target to be tracked is the delivery robot which is tasked through the infrastructure to travel to locations in the corridors within the tracking system's coverage.

The infrastructure operates as follows. The systems connect to the infrastructure, register their services and wait for requests. The users connect to the infrastructure and receive the global list of services, consisting of the three tracking services for the differently colored targets and one delivery service. The users select whichever service they desire and are connected to the applicable system. They then have a one-to-one interaction for each system they are connected to. Additionally, they retain access to the global service list for many-to-many interaction.

The experiment ran for approximately 50 minutes with 10 users connecting to the infrastructure and periodically requesting and terminating services. Figure 4 shows the server's performance during the experiment. The data shown represents the time for the server to process service request, list request, and service termination messages. The instantaneous number of online users is also shown. As can be seen, the number of users has little effect on the response times. The main reason for this is that the infrastructure's operations are event-based on the user's side. Therefore, connected users require no maintenance when there is no activity. On the system side, a ping signal is sent from the server every 20 seconds to ensure all systems connected to it are still serviceable. A system has 10 seconds to respond before it and its services are removed from the server.

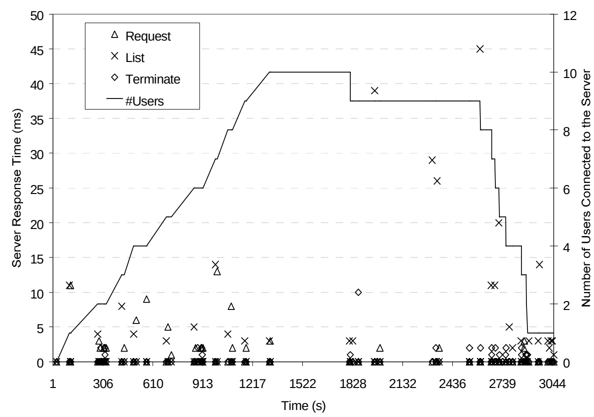

Figure 4. Server response times for Experiment 1. The number of users connected at any point in time is represented by the solid line and the right y axis. The server's response times to the various messages are shown by the symbols and correlate to the left y axis.

While the server's response times to requests provide evidence of its efficiency, the response times from the user's perspective provides a better indication to the efficiency of the infrastructure. In addition to the time taken by the server to process the requests, the communication delays between infrastructure layers, system client processing times and UDP packet queue processing can also contribute to these response times.

Table 1 lists the statistics pertaining to requests for services and service lists. These constitute the two most time-intensive messages used in the infrastructure. The times are measured from when the user made the request to when a response was received which is either from the server if the request is refused or from the system to commence interaction.

Table 1. Statistics from the user's perspective

	Service List Requests	Service Requests
Number of Requests	30	19
Maximum Response Time	170 ms	64 s
Median Response Time	43 ms	691 ms
Minimum Response Time	5 ms	38 ms

Requests for the global service list from the server are generally carried out quickly since the transaction involves only the user clients and the server. Requests for services take longer since they are processed by the server and the system clients. Apart from the transition times, it may take some time for the system client to respond to the user's request depending on its ongoing activities. For the tracking system, one robot was responsible for managing all users' requests in addition to its tracking duties. As a consequence of these duties and the limited processing capability and memory on the robot, the response times sometimes took several seconds.

The statistics for the service requests are listed in Table 2. The users selected 19 tasks, 14 of which were granted. Three tracking service requests were refused due to the maximum quota of 10 simultaneous requests being exceeded[1]. The system client notified the users in these cases. Two request failures occurred due to the robot being overburdened with too many requests to manage simultaneously. Although the maximum number of requests that the system client manages was set to 10, the maximum number the robot's internal system could reasonably sustain while carrying out its part as a tracker was eight. During peak activity, eight users were being managed by the tracking system for more than 20 minutes, providing updates at 1 Hz.

Table 2. Service statistics showing the number of tasks selected and service running times

	Delivery Service	Tracking Service
Number of Granted Tasks	3/3	11/16
Maximum Time	1754 s	2057 s
Median Time	341 s	1478 s
Minimum Time	185 s	2 s

Experiment 2 – Service Linking

The second experiment demonstrates the ability of the infrastructure to link heterogeneous services to provide a higher-level function to the user. It is similar to our previous service-linking experiment [17] except it is carried out in a real building environment rather than simulated open space.

In this experiment, both tracking and delivery systems are used. The tracking system provides users with the locations of particular colored targets whereas the delivery robot requires a location to deliver to. By linking these semantically similar services, a dynamic location delivery function is created. In this scenario, the delivery target is a person wandering around the environment.

[1] Even though there were only 10 users, a user could request more than one task from a system if the system allows it.

The robots are configured the same as Experiment 1, except one of the tracking robots is now mobile (T3 in Figure 3). Although this robot can traverse the whole environment while searching and tracking, it is confined to the corridor where the stationary trackers are located.

Once the systems and the infrastructure are connected, a user logs on and selects the target tracking service to determine whether the target is currently being tracked. Once it is, the user selects the linking option from the service menu, which prompts for the data-providing service and the command service. These are selected as the applicable target tracking and delivery service respectively.

The operation of the infrastructure at this point is the same as if two separate services were selected. Both connect to the user client, which checks if the data service is part of a link. Since it is in this case, the information is sent to the command service's system client as well as being displayed to the user. The command service's system client parses the received information for the goal location. This effectively automates the user's manual goal location entry. Consequently, the delivery robot is constantly provided with the updated target location. In the cases when the target is lost or there is no detected location information in the data sent to the delivery robot, it proceeds to the last provided location. This helps deal with temporary communication failures and target loss. Once the delivery robot is within a meter of the goal location, it halts and sends a message to the user that it is at its commanded location.

In this experiment, the tracking information is sometimes provided by the mobile tracking robot, which resulted in a two-robot convoy following the target. This application of a linked service could easily be extended to generate a leader-follower behavior for multiple robots if they need to be relocated *en masse*.

5 Conclusions

In the context of developing a scalable human-robot interaction mechanism, we have assumed that the systems involved have some degree of autonomy. This assumption allows a general and simple mechanism to be developed. Our interaction infrastructure follows this paradigm and allows users to interact with systems at two levels: many-to-many and one-to-one. The many-to-many level is required to expose the tasks or robots' services to the humans, and upon selection the users have more personal one-to-one interaction. These levels are expected to be required in large scale robot applications such as assembling solar panels in space or robot-assisted search for casualties after an earthquake. The experiments presented in this paper demonstrate the infrastructure's application towards these types of applications.

A future step for enhancing the infrastructure is to distribute the server to provide greater redundancy. This will allow more robust operation in the presence of either communication or server failures. Currently, if the server fails, the infrastructure is unusable for new interactions, however any existing interaction between systems and users remains unaffected.

Establishing the potential numbers of systems and users the infrastructure can support is important but logistically difficult with real humans and systems. It can however, be analytically determined by developing a queuing model from the infrastructure's performance. Results obtained from the model can then be correlated in simulation.

Acknowledgements

This work is supported in part by the DARPA Mobile Autonomous Robot Software (MARS) Program under grants DABT63-99-1-0015 and 5-39509-A (via UPenn), the DARPA Software for Distributed Robotics (SDR) Program under contract 4400057784 (via SAIC), and the ONR DURIP program under grant 00014-00-1-0638.

Bibliography

[1] A. Agah, "Human Interactions with Intelligent Systems: Research Taxonomy", *Computers and Electrical Engineering*, 2001, no. 27, pp. 71-107.

[2] K. Ali, and R. Arkin, "Integration of Reactive and Telerobotic Control in Multi-agent Robotics Systems", Proceedings, *From Animals to Animats 3: Simulation of Adaptive Behavior 1994*, pp. 473-478, MIT Press.

[3] R. Arkin, T. Collins, and Y. Endo, "Tactical Mobile Robot Mission Specification and Execution", Proceedings, *SPIE-99, Mobile Robots XIV,* Boston, MA, September, 1999, pp. 150-163.

[4] I. Elhajj, J. Tan, N. Xi, W. Fung, Y. Liu, T. Kaga, Y. Hasegawa, and T. Fukuda, "Multi-site Internet-based Cooperative Control of Robotic Operations", *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 2000, pp.826-831.

[5] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd, "Integrating Information Appliances into an Interactive Workspace", *IEEE Computer Graphics and Applications,* May/June, 2000, pp. 54-65.

[6] B. Gerkey, R. Vaughan, K. Støy, A. Howard, G. Sukhatme, and M. Matarić, "Most Valuable Player: A Robot Device Server for Distributed Control", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, Wailea, Hawaii, Oct. 29-Nov. 3, 2001, pp. 1226-1231.

[7] S. Goldsmith, S. Spires, and L. Phillips, "Object Frameworks for Agent System Development", Sandia National Laboratories, Tech. Rep. SAND98-0742A, 1998.

[8] H. Jones, and M. Snyder, M. "Supervisory Control of Multiple Robots Based on a Real-time Strategy Game Interaction Paradigm", *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*, Tucson, AZ, Oct. 2001, pp. 383-388.

[9] B. Jung, and G. Sukhatme, "A Region-based Approach for Cooperative Multi-Target Tracking in a Structured Environment," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, Sep. 30-Oct. 4, pp. 2764-2769.

[10] J. Lee, and H. Hashimoto, "Intelligent Space - Its concept and contents", *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Takamatsu, Japan, 2000, vol. 2. pp. 1358-1363.

[11] R. Murphy, and E. Rogers, "Cooperative Assistance for Remote Robot Supervision", *Presence,* Special Issue, vol. 5, no. 2, pp. 224-240, 1996.

[12] L. Phillips, S. Goldsmith, and S. Spires, "CHI: A General Agent Communication Framework", Sandia National Laboratories, Tech. Rep. SAND98-2825C, 1998.

[13] S. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments", *Proc. Ubiquitous Computing Conference*. *Lecture Notes in Computer Science*", vol. 2201, 2001, pp. 56-75.

[14] P. Rybski, I. Burt, T. Dahlin, M. Gini, D. Hougen, D. Krantz, F. Nageotte, N. Papanikolopoulos, and S. Stoeter, "System Architecture for Versatile Autonomous and Teleoperated Control of Multiple Miniature Robots", *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001.

[15] H. Surmann, and M. Theißinger, "ROBODIS: A Dispatching System for Multiple Autonomous Service Robots", *Proceedings of the FSR'99 Robotics Applications for the Next Millenium,* Pittsburgh, PA, 1999, pp. 168-173.

[16] A. Tews, G. Sukhatme, and M. Matarić, "An Infrastructure for Large-Scale Human-Robot Interaction", Technical Report CRES-02-001, Center for Robotic and Embedded Systems, University of Southern California, August, 2002.

[17] A. Tews, M. Matarić, G. Sukhatme, and B. Gerkey, "G'day Mate. Let me Introduce you to Everyone: An Infrastructure for Scalable Human-System Interaction", Technical Report CRES-02-004, Center for Robotic and Embedded Systems, University of Southern California, August, 2002.