G'day Mate. Let me Introduce you to Everyone: An Infrastructure for Scalable Human-System Interaction

Ashley D. Tews, Maja J. Matarić, Gaurav S. Sukhatme, Brian P. Gerkey

Robotics Laboratory, University of Southern California, Los Angeles, USA, {atews/ mataric/gaurav/gerkey}@robotics.usc.edu

Abstract

We are exposed to physical and virtual systems every day. They consist of computers, PDAs, wireless devices and increasingly, robots. Each provides services to individual or groups of users whether they are local or remote to the system. Services offered by these systems may be useful beyond these users to others, however connecting many of these systems to more users presents a challenging problem. The primary goal of the research presented in this paper is to demonstrate a scalable approach for connecting multiple users to the services provided by multiple systems. Such an approach must be simple, robust and general to contend with the heterogeneous capabilities of the services. An infrastructure is presented that addresses these scalability requirements and establishes the foundation for contending with heterogeneous services. Additionally, it allows services to be linked to form higher-level abstractions. The infrastructure is demonstrated in simulation on several similar multirobot systems with multiple users. The results propose it as a solution for large-scale human-system interaction.

1. Introduction

The issue of scalability in human-system interaction applies to both the human and the system ends. How do many people interact with many systems? A general solution for interacting with a small number or single systems is to maintain interaction infrastructures dedicated to a specific set of users. Such approaches tend to be application-specific and are scalable only in their local operating environment. There are many issues to address if the infrastructure is to allow users outside of these bounds, as well as extending the number of managed systems. The main issues concern the scalability of the infrastructure, the heterogeneity of the information provided by other systems, and the user interfaces required for meaningful interaction with these systems. There is a large amount of diversity in each of these issues considering the user can have varying levels of interaction with a system. For example, a user teleoperating a robot requires a more specific user interface and higher communication bandwidth than a supervisory user of a multi-robot system. Developing an infrastructure that can address these issues and their diversities is the focus of this paper.

A system is an entity that provides services (assistance/information) to a user. A service is thus a system resource. A user is defined as a human or system requiring the service. The requirement of connecting large numbers of systems to large numbers of users demands a scalable and simple solution. It is also beneficial for existing systems to undergo minimal modification to allow them to interface with such a solution. The approach must therefore be able to:

- scale to large numbers of users,
- scale to large numbers of systems and their services,
- manage the diversity of the information provided by the services, and
- allow for the variety of user interfaces required for the services.

The infrastructure presented in this paper has been developed under consideration of all issues but so far primarily addresses the first two. Additionally, it extends the usability of the infrastructure allowing services to be linked to form higher-level abstractions. This is beneficial for sequential and concurrent control of multiple services and adds more flexibility to the infrastructure. Sequential control allows one service to provide an event-based trigger to initiate another. Concurrent control can be provided in some instances when the services have semantically similar information: one service provides the input to another for their concurrent operation. An example is demonstrated in this paper where an object-tracking service provides the dynamic location of a robot to a package delivery robot that uses it as a navigation goal.

The following section provides a review of the research in related areas of human-system interaction. Section 3 presents the infrastructure. Section 4 presents experimental results that demonstrate some of the capabilities of the infrastructure. Section 5 provides a discussion of the infrastructure components. Section 6 provides conclusions based on the experimental results in terms of the requirements of the infrastructure described above. Section 7 identifies the goals of the next development stage of the infrastructure.

2. Related Work

Considering a system can consist of physical entities (such as robots or embedded devices) and virtual entities (such as the services provided by computer programs), human-system interaction covers large areas of research. In [1], a taxonomy is presented categorising human and intelligent system research. The concept of an intelligent system is a system that allows for user interaction (e.g. tele-robotics). The categories include the application of the research, the research approach, system autonomy, interaction distance and interaction media with each category divided into subcategories. Many references are grouped under these categories, which illustrates the vastness of the field. Another axis that can be added to the classifications is the scalability of the approaches in terms of users and systems.

This section classifies human-system architectures in terms of their scalability assuming a system consists of a set of nodes that provide a set of services to users. These classifications are shown in Table 1 and are discussed in the following sections that separate them into single (non-shaded table entries) and multi-user architectures (shaded table entries).

Table 1: U	User/system s	calability o	classification
------------	---------------	--------------	----------------

	Users	Systems
1	one	one
2	one	multiple
3	multiple	one
4	multiple	multiple

Single-user System Architectures

There are many examples relating to the entries in the table above. The issues concerning scalability are highlighted when examining these examples. An important issue that affects the scalability of an approach is how tightly coupled the users are with the system. This concerns access to system resources and levels of control over system services. Single-user systems allow the user full access to the system services and control. Scaling these to more users can be difficult due to these aspects (such as communication bandwidth) having to be shared. Tele-operation of a single system node is an example where this is highlighted [12]. Multi-robot research also provides examples where multiple robots can be controlled individually or as a group by a single user [2][3][11][16]. In each of these systems, the nodes have a certain degree of autonomy and the user has either high level group control, or low level individual control. This indicates that the higher the number of users an interaction architecture has to deal with, the looser the coupling between users and systems.

The concept of a single user for multiple systems indicates a hierarchical command structure where there is a top-level entity that controls the systems below their level.

Multi-user System Architectures

Multi-user and multi-service systems are forced to address scalability, system resource allocation and control. A key concern is how to control the information and communication pathways for multiple users. Common approaches involve using multi-agent architectures [6][10][14][15] and the Internet.

Multi-agent architectures offer services to users with suitable access to the systems they serve. Examples include interactive workspaces [7][13] which consist of dedicated environments with embedded devices providing services for users. The users generally must be in the environment to access the services.

Multi-user/multi-robot architectures are demonstrated by [5][17]. These systems use the Internet as a communication medium for commanding robots from remote locations.

Many of the examples presented in this section are scalable only in their local operating environment or application domain. The infrastructure presented in the next section is intended to operate beyond these limits, as well as contending with the diversity of coupling and resource sharing required by individual systems.

3. Infrastructure

The infrastructure we have developed is shown in Figure 1 as the shaded boxes. It is a client/server method using UDP as the communication protocol. Each user has a client that provides an interface with the infrastructure. Similarly, the system client acts as the system interface to the infrastructure. The Service List Server manages the initial contact between users and systems. Once a user has selected their desired services, the server connects the associated user's and system's clients, then removes itself from their communication loop.

To be scalable, the infrastructure must be general, simple and initially provide a loose coupling between users and systems. It has to be general to manage the heterogeneous services provided by unknown systems. It has to be simple to reduce any bottlenecks that can occur, particularly by the communication paths. It also must require as little non-invasive addition as possible to existing systems for it to be accepted. These goals constitute the philosophy for infrastructure design. For



Figure 1: The infrastructure for connecting multiple users to multiple system services

the implementation described in this paper, the following assumptions are made with the presumption that those that present limitations will be addressed in subsequent development phases:

- 1. The systems already exist.
- 2. The contact details (i.e. host and port number) of the server are known by system clients.
- 3. The user interfaces for each system are simple and have low functionality.
- 4. Each system can provide multiple services, but only one can be linked to a user for a session.
- 5. The semantics of the interaction are understood by the user and the system so that system responses to user requests are what the user expects.
- 6. System managers are motivated to have their system's services available to users.

The Infrastructure's Sequence of Operation

This section details the steps for using the infrastructure.

Step 1. Generating the global service list

The first step for using the infrastructure is to compile the complete list of services offered by the systems. Any time a system connects to the server, its client sends its list of services. The Service List Server stores these descriptions along with the system clients' contact details.

Step 2. Presenting the list to the user client

Whenever a user client connects to the server, the server requests the service lists from all known system clients. This ensures all listed services are current and valid. The global list is then sent to the requesting user client¹. The list is presented to the user as a menu along with an

additional option of combining compatible services. Service compatibility must fulfil two criteria. First, the services must be semantically compatible so the shared information has a uniform meaning. Second, one service must be a data provider for the other. This is established by labeling each service as a data provider or a command service as part of the service descriptions.

Step 3. Selecting services

When a user selects a service, the user client sends the request to the server which forwards it to the relevant system along with contact details for the user client. The server then disconnects from both the user's and system's clients allowing direct and higher bandwidth between them.

Step 4. Using the service

The user is presented with a new menu of actions for the requested service. These include terminating the service and establishing a new connection to the server (transparent to the user), exiting from the infrastructure, or starting the service. Once the user selects the option for starting the service, a signal is sent to the system client to start executing the service. In future infrastructure development, this will also establish a service-specific user interface. The user interacts with the selected system until the service completes.

4. Validating the Operation of the Infrastructure

The main goals of the infrastructure are to allow heterogeneous services, large numbers of users and systems, and to provide a mechanism for linking services, if appropriate, with little overhead to existing systems. To illustrate how the infrastructure achieves many of these goals, two simulation experiments are

¹ In future infrastructure development, this will refresh all user clients' lists.

carried out using the Player/Stage² platform [9]. The Player platform provides a library of device software for developing Pioneer 2 mobile robot applications. The Stage simulator interfaces with the Player software to provide valid real system operation of Player devices. It also allows multiple robots to be simulated in userdesigned environments in real or accelerated time. In our experiments, the environment is a simulated large room.

The systems used in the experiments are based on the Murdoch [8] task allocation system. Murdoch uses an auction-based approach to task allocation. When a task is to be assigned, Murdoch holds a first-price auction and each available robot submits a bid that represents its fitness for the task; the fittest robot is then awarded the task. The version of the system used in our experiments allows three tasks to be carried out, annotated with their service type:

- object-tracking (data service) where a robot tracks an object with a coloured marker on it and report its position,
- random-walk (data service) where a robot randomly moves around the environment and reports its position, and
- *goto_xy* (command service) where the robot navigates to the location provided.

Each task is designated as a data or command service as mentioned previously to allow for linking options. Data services can be considered as providing outputs while command services require inputs. A system client interface is added to each instance of Murdoch. The client has the function of handling server requests, requesting a service from the system once a user has made a selection, and providing the user client with the service-specific user interface.

In the following experiments, a system consists of one robot that can provide the three services. The choice of a single shared environment and multiple instantiations of the same system provide a focus for analysing the issues of scalability and service linking. Once this is established, the next development phase of the system will be to test it with multiple different systems.

Experiment 1 - Service Linking

The first experiment demonstrates the service linking ability of the infrastructure by simulating a usertracking and package-delivery task. This is synonymous with recruiting a robot to deliver a package to a nonstationary person. The location of the person can be found by employing a user-tracking service that a sensor network can provide. In this experiment, the persontracker is an object-tracking robot. The user gives the delivery robot the package, then selects and links the *object-tracking* and package delivery (*goto_xy*) services from the two independent systems. Once the target person has been located, position information is sent from the tracking robot to the delivery robot so it can navigate to the person and deliver the package.

Setup

In this experiment, a randomly wandering robot is used to represent the target person in the environment. The robot has a coloured marker for identification. The *object-tracking* robot searches for the marker and provides position coordinates of the tracked object. The *goto_xy* robot is considered to have a package given to it by the user requesting the service. Since both of these robots are operating in the same environment and with the same type of system, the semantics of the data sent between them are preserved.

Operation

The Service List Server and the two systems are initiated. Since each has three services, the Service List Server receives descriptions for six services. There is only a single user for this experiment so a User Client is started and displays the six choices along with the service linking option. The descriptions identify to the user that there are two independent systems in the same local environment with three services each. The user selects the service linking option, along with the *object-tracking* service from one system and *goto_xy* service from the other. The server sends messages to the two system clients to contact the user client and then removes itself from their communication path.

The user client waits to be contacted by the system clients. The user is then presented with the service-specific interface options and initiates the services. This sends 'start' signals to the system clients. The *object-tracking* robot starts searching for the person in the environment while the *goto_xy* robot waits for locations to be sent to it. Once the person is found, their location information is sent to the user client which forwards it to the *goto_xy* service. The robot providing the *goto_xy* service navigates towards the provided location. Once it has intercepted the person, the simulation ends as the package has been delivered.

Experiment 2 - Scalability Testing

The purpose of this experiment is to provide an indication of the scalability of the infrastructure. It is a partial test in the sense that it does not test the infrastructure to its performance limits. Instead, a

² Player/Stage was developed jointly at the USC Robotics Research Labs and HRL Labs and is freely available under the GNU General Public License from http://playerstage.sourceforge.net

theoretical analysis of these limits is presented followed by an experiment with ten users and systems.

To theoretically determine the scalability limits of the infrastructure, its phases of operation can be analysed. The two phases are 1) compiling and supplying the global service list, and 2) providing the connection between the user and the selected services. The second phase is simple. Once the user is connected to the service, there is no overhead to the infrastructure since the user and system clients are operating remotely to it.

The first phase consists of many users, a single server, and many systems. Analytically, the scalable limit can be determined by the amount of information stored by the server and the communication bandwidth. Each service consists of a single-line description (256 bytes) which is stored on the server. Also stored is a list of all known system client addresses (4 bytes each) and ports (2 bytes each) so they can be sent requests for updates. The total overhead for system information storage on the server is less than 260 bytes for each service since a system client with multiple services will only have one instance of its address and port number stored. This allows for a large number of services to be stored.

The infrastructure communication is handled by UDP sockets, which have more of an impact on scalability. Each system and user adds a communication path to the server. These paths are only active upon a user's menu selection, or a new service or user addition. The maximum throughput of activity occurs when the service list is requested. Service list requests occur when a new user connects to the system or a service list refresh signal is received from a user client.

The size of each communication packet (one service description plus UDP header information) is approximately 270 bytes. There is a linear relationship between the number of users and the number of services for the maximum throughput calculation. If there are 100 services and the bandwidth of the network is 10Mbits/sec, the approximate theoretical user limit of the system is 46 users. This will only occur if they have all simultaneously logged on or requested a service list refresh which is highly unlikely. This feasibly indicates the infrastructure is scalable well beyond this theoretical limit.

Setup

The partial scalability experiment simulates ten users and ten systems with each system providing three services for one robot. Therefore, there are ten robots in the environment. Each user selects a service and the infrastructure is observed to determine if there are any shortcomings with connecting multiple users to multiple systems' services.

Operation

As the systems come online, their services' descriptions are sent to the Service List Server. As the users come online, they are presented with this list of services and can select whichever they require. Systems and users can connect in any order and at any time. Each user selects a data service for this initial phase of the experiment.

There were no observed problems with connecting this many users to services, so to further test the versatility of the infrastructure, a user requested a linked service³. In the presence of the other services and robots in operation, the user selected a *random-walk* service for the data service and linked the information to a *goto_xy* service on another system. The position of the randomly walking robot was forwarded to the *goto_xy* robot which made it follow the other robot around. This was carried out without any problems. The other users stopped and started services dynamically throughout the simulation and the system was robust to these requests.

5. Discussion

Throughout this paper, it is stated that the infrastructure needs to be general and simple with little implementation overhead if it is to be scalable. This section describes how three primary components of our infrastructure achieve these goals.

Scalability: The Service List Server

The main reason for using a centralised server to gather the service list is its simplicity and to reduce the communication flow between the *m* possible users and the *n* possible systems. This reduces the communication problem from magnitude $(m \ge n)$ if they were fully interconnected, to (m + n). The risk with this approach is that a loss of the server affects service selection. There are many possible mechanisms to overcome this limitation that have not been explored yet, such as distributing and linking servers with partial service lists.

Failure of the server does not affect already connected interactions which also benefits scalability.

Low Overhead: The System Clients

The purpose of system clients is to provide an interface to a system's services with as little implementation overhead as possible. The overhead in this case is the modification required for each system to allow it to interface with the infrastructure. This mainly consists of the UDP client message-handling software with servicespecific input and output data managers. For the

³ Two users had stopped using a system allowing two robots to be waiting for service requests.

systems used in our experiments, this proved to be trivial to implement.

Generality: The User Clients

The user clients present the global service list to their user initially and then the service-specific interface. Since the type of information they are required to manage is unknown, a general solution is required. This is established by allowing the user interface to be determined by the system client. For the research presented in this paper, the user interface is simple and does not provide a great deal of functionality.

The user client also has the ability to combine services. This requires a semantic understanding of the data between the system and user clients since combining services redirects the information supplied by one service to the input of another. For the current implementation, this is the user's responsibility. Incompatible data may result in the command system client not responding to the data input. While this may not be a problem for discrete signals (such as could be used for sequential control), it can be a problem for more context-rich data such as location information. To overcome this possible limitation, a protocol can be used to identify semantically similar command and data services. This allows systems with the same type of data to have their services linked as demonstrated in the experiments.

6. Conclusions

The primary goal of the infrastructure presented in this paper is to demonstrate a scalable solution for interfacing multiple users with multiple systems. The infrastructure is simple, provides a loose initial coupling between users and systems, and requires minimal modification to existing systems desiring to add their services. In addition, it allows services to be combined by the user to develop higher level abstractions.

Due to the simple architecture of the infrastructure, various existing and proven technologies can be applied to enhance its operational effectiveness. For example, a scheduling component can be added to manage multiple requests for the same service, and web browsers can be used for the user interface.

The results demonstrated by the initial experiments indicate that the infrastructure promises high scalability. There are many enhancements for the next iteration of development which are described in the next section.

7. Future Work

The main enhancements left for the next version of the infrastructure are:

- allow the system and user clients to handle more diverse services,
- allow the service linking option to link many services such as providing multiple *goto_xy* robots with the same location information from another service,
- add the mechanism to generate higher functionality service-specific user interfaces [15] once the user has selected the service, or allow the users to compile their own interface depending on the available modalities [4],
- decentralise the server so it has redundancy with minimal impact on performance, and
- allow parameters to the services such as time for completion.

The long-term goal for the infrastructure's development is to remove the concept of a user and system by making them all peer entities. This will allow users to request and provide services in the same way as a system does. Any service request can then be provided by an anonymous entity.

8. References

- 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", *From Animals to Animats 3: SAB 1994*, pp. 473-478.
- [3] R. Arkin, T. Collins, and Y. Endo, "Tactical Mobile Robot Mission Specification and Execution", *Mobile Robots XIV*, Boston, MA, September, 1999, pp. 150-163.
- [4] B. Dorohonceanu, B. Sletterink, and I. Marsic, "A Novel User Interface for Group Collaboration", *Proceedings of* the 33rd Hawaii International Conference on System Sciences, Maui, Jan. 4-7, 2000.
- [5] 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.
- [6] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "KQML as an Agent Communication Language", *Proceedings of the Third International Conference on Information and Knowledge Management*, ACM Press, 1994, pp. 456-463.
- [7] 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.

- [8] B. Gerkey, and M. Matarić, "Principled Communication for Dynamic Multi-Robot Task Allocation", *Experimental Robotics VII, LCNIS 271*, Springer-Verlag, 2001, pp. 353-362.
- [9] 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.
- [10] S. Goldsmith, S. Spires, and L. Phillips, "Object Frameworks for Agent System Development", technical report no. SAND98-0742A, Sandia National Laboratories, 1998.
- [11] 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, October 2001, pp. 383-388.
- [12] K. Kawamura, R. Peters, C. Johnson, P. Nilas, P. and S. Thongchai, "Supervisory Control of Mobile Robots Using Sensory Egosphere", *IEEE Symposium on Computational Intelligence in Robotics and Automation*, Jul. 29-Aug. 01, Banff, Canada, 2001.
- [13] 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.
- [14] L. Phillips, S. Goldsmith, and S. Spires, "CHI: A General Agent Communication Framework", technical report no. SAND98-2825C, Sandia National Laboratories, 1998.
- [15] S. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments", *Proceedings of the Ubiquitous Computing Conference. Lecture Notes in Computer Science*", vol. 2201, 2001, pp. 56-75.
- [16] 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.
- [17] 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, pp. 168-173.