Flexible Communication in Multi-robotic Control System Using HEAD: Hybrid Event-driven Architecture on D-Bus ⋆

Md Omar Faruque Sarker and Torbjørn S. Dahl.

Robotic Intelligence Lab, University of Wales, Newport
Allt-yr-yn Campus, Allt-yr-yn Avenue, Newport, NP205DA, UK
(e-mail: Mdomarfaruque.Sarker—Torbjorn.Dahl@newport.ac.uk)

Abstract: Direct real-time communication among various software components of a multi-robot system (MRS) is much more complicated than that in software simulations. Existing inter-process communication (IPC) mechanisms, such as pipes, shared memory etc., are very rigid and usually enforce tight coupling among software components. Thus they do not integrate well with heterogeneous multi-robot control applications of a relatively larger MRS that typically consists of tens of robots and various sensing and monitoring elements interconnected through several host PCs. In this paper, we present a modular, flexible and decentralized multi-robot control architecture, namely hybrid event-driven architecture on D-Bus (HEAD), that overcomes these issues by decoupling IPC through D-Bus. D-Bus is a relatively newer IPC technology for modern Linux desktop environments. It typically uses a message bus daemon that facilitates asynchronous data sharing among multiple processes. Here, we show that by using only a single type of message, namely D-Bus *signal* type, HEAD can efficiently enable real-time interactions among heterogeneous multi-robot control applications. The design of HEAD is flexible enough to add various types of existing and new software components with minimum programming effort. As an example, we present how we achieve a decentralized peer-to-peer communication behaviours among robot controller clients by simply adding only a few lines of new code leaving the major IPC implementation intact. This paper also reports the performance of D-Bus, under both constant and variable IPC load, obtained from our MRS implementation of a manufacturing shop-floor scenario with 16 e-puck robots.

Keywords: multi-robot system, real-time control, inter-process communication, D-Bus

1. INTRODUCTION

Inter-process communications (IPC) among various desktop software components enable them to talk to each other and exchange data, messages or request of services. Technological advancements in computer and communication systems now allow robotic researchers to set-up and conduct experiments on multi-robot systems (MRS) from desktop PCs. Many compelling reasons, including open licensing model, availability of open-source tools for almost free of cost, community support etc., make Linux as the most preferable operating system for MRS research. However the integration of heterogeneous software components in Linux desktop becomes a challenging issue, particularly when each robot-control software (hereafter called robot-controller client or RCC for short) needs sensory and other data input from various other software components (e.g. pose data from a tracker server, task information from a task server etc.). Traditional IPC solutions in a standard Linux desktop, e.g. pipes, sockets, X atoms, shared memory, temporary files etc. (hereafter called *traditional IPCs*), are too heterogeneous to meet the demand of a dynamic software system **?**. On the other hand, complex and heavy IPC like CORBA fails to integrate into a development tool-chain efficiently. They also require a steep learning curve due to their complex implementations. Besides, the failure of Desktop Communication Protocol (DCOP) in system-wide integration and interoperability encouraged the development of the D-Bus message bus system (D-Bus for short).

D-BUS was designed from scratch to replace CORBA and DCOP and to fulfil the needs of a modern Linux system. D-BUS can perform basic application IPC, allowing one process to shuttle data to another. D-BUS can facilitate sending events, or signals, through the system, allowing different components in the system to communicate and ultimately to integrate better. D-BUS is unique from other IPC mechanisms in several ways, e.g. 1) the basic unit of IPC in D-BUS is a message, not a byte stream, 2) D-BUS is bus-based and 3) It has separate system-wide and user/session-wide bus **?** . The simplest form of communication is process to process. D-BUS, however, provides a daemon, known as the message bus daemon, that routes messages between processes on a specific bus. In this fashion, a bus topology is formed, allowing processes to speak to one or more applications at the same

⋆ This research has been funded by the Engineering and Physical Sciences Research Council (EPSRC), UK, grant reference EP/E061915/1.

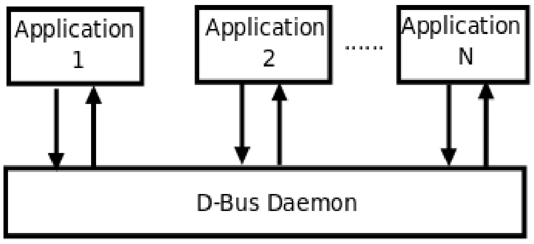

Fig. 1. A typical view of D-Bus message bus system

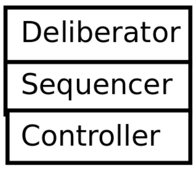

Fig. 2. Classical three layer robot control architecture **?**

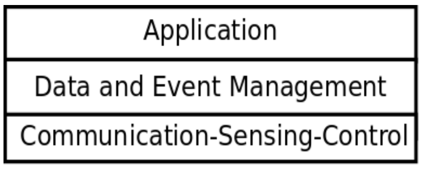

Fig. 3. Our abstract multi-robot control architecture

time. Applications can send to or listen for various events on the bus.

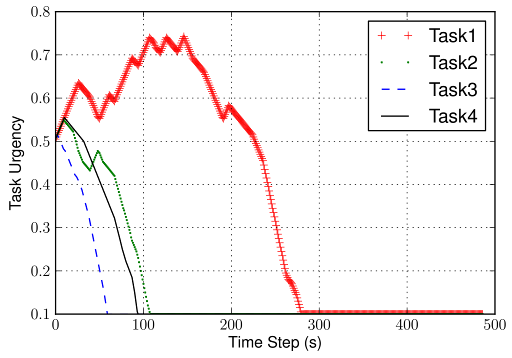

Fig. 6. Dynamic changes in task urgencies

2. D-BUS COMMUNICATION MODEL

2.1 D-Bus Overview

2.2 Strategies for Application Integration

3. HYBRID EVENT-DRIVEN ARCHITECTURE ON D-BUS (HEAD)

3.1 Overview of Multi-robot Control

3.2 Characteristics of HEAD

3.3 D-Bus Overview

3.4 D-Bus Signal Interfaces

3.5 Common Interfaces

3.6 Integration of Software Components

4. FLEXIBLE COMMUNICATION SCHEMES OF *HEAD*

4.1 Centralized Communication

4.2 Decentralized Communication

5. IMPLEMENTATION OF A MULTI-ROBOT MANUFACTURING SCENARIO

5.1 Characteristics of HEAD

5.2 D-Bus Overview

5.3 Multi-robot Task Allocation

5.4 Experiment Setup

5.5 Results and Discussions

6. CONCLUSION

Appendix A. D-BUS LISTENER CODE

Appendix B. D-BUS EMITTER CODE

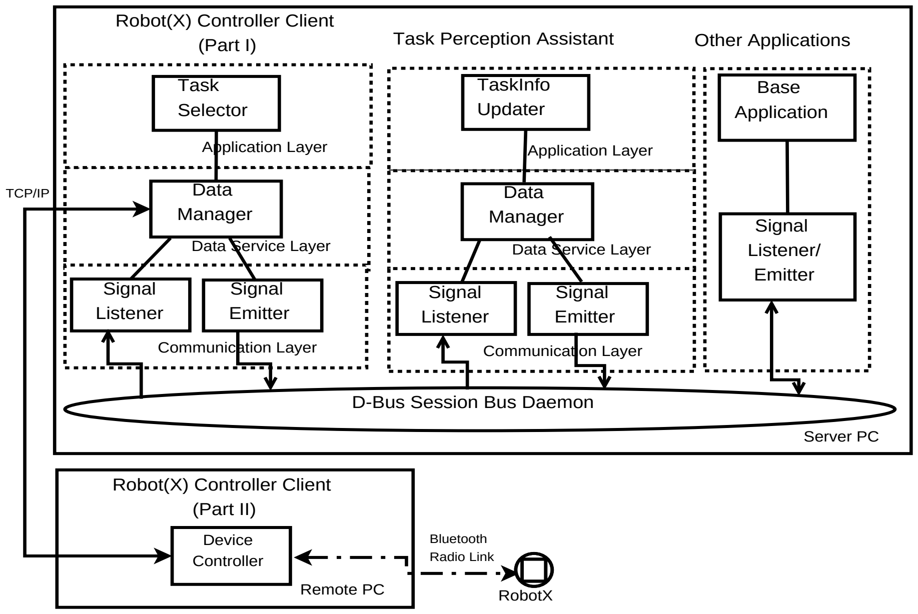

Fig. 4. General outline of *HEAD*. Robot-Controller-Client application has been splitted into two parts: one runs locally in server PC and another runs remotely, e.g., in embedded PC

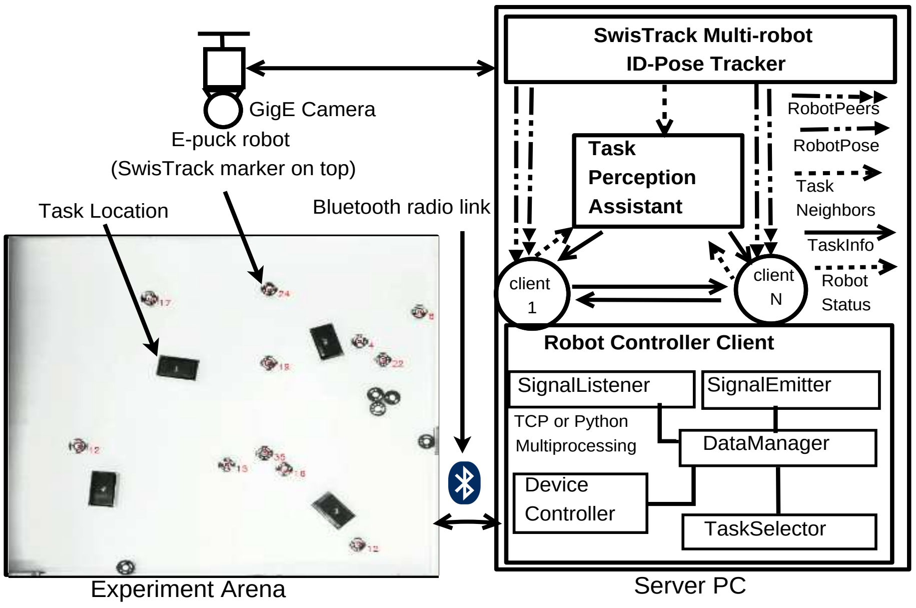

Fig. 5. An example implementation of *HEAD*. Robot-Controller-Clients can talk to each other and exchange task information through D-Bus signals

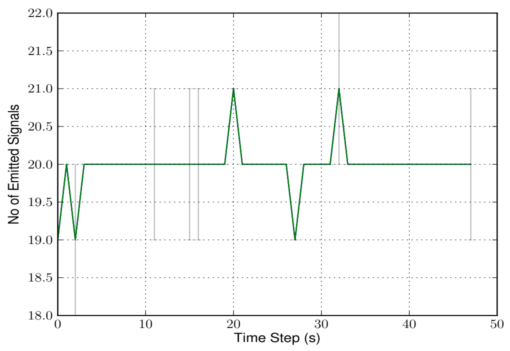

Fig. 7. D-Bus task information signalling frequency of TPA in centralized communication mode

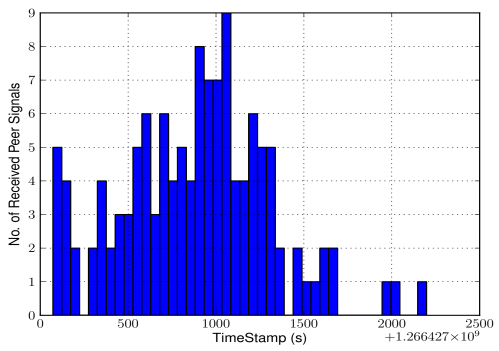

Fig. 8. Robot12's simultaneous reception of task information signals from peers

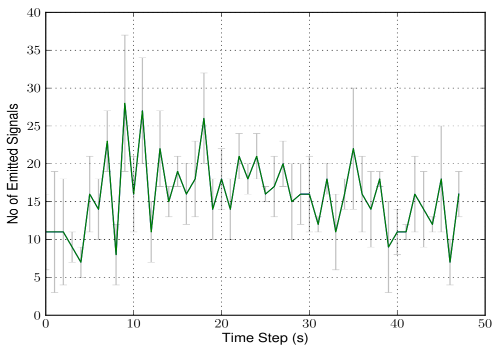

Fig. 9. Local P2P task information signalling frequency of all robots in decentralized communication mode