Task Administration via Peer-to-Peer Communication

Yasin Sahin*, Sven Oberwalder, Fabian Hitzenberger, Karl Dopplinger, Raphael Ackerl, Julian Kerer
Department of Computer Science
HTL Wiener Neustadt
Wiener Neustadt, Austria
*Corresponding author email: sahin.yasin@student.htlwrn.ac.at

***Abstract*—Time is a very substantial aspect both in the Botball competition and in various automation tasks. The speed at which the machines operate, and thus the efficiency, can be increased by a both task and time optimized cooperation between them. This work proposes a prototype of a novel library called Task Administration Data Exchange Library (TADElib) which provides task coordination between robots, machines, and other types of devices. Furthermore, the analysis of conducted experiments, which are designed within the Botball environment, shows the increase in efficiency.**

***Index Terms*—task, administration, communication, multi-robot**

I. Introduction

In most scenarios one robot alone could be sufficient to complete a required task. However, having more robots working together may have more advantages, particularly when the task contains more than one objective, similar to the Botball competition. For example, the task can be split up among the robots, so that the task can be finished more quickly [1]. The main problem of working with multiple robots is that they must be coordinated in advance and work in a coherent manner. Otherwise, they might run into each other, trying to accomplish the same job, or hinder each other in a different way. While this problem can be solved statically, a dynamic task allocation has way more benefits:

It is possible that there could be the need to temporarily pause a task, if, for example, one of the robots detects that another robot may interfere with its current task. Additionally, through effective management, one can detect whether the output of accomplished tasks is sufficient, or a plan B should be executed. Another benefit could be, the possibility of perceiving execution errors and make other participants of the system help out the machine in need, if the library is supported by various algorithms.

For the reasons mentioned above, this study aimed to develop a library that serves the purpose of handling the communication and task management between participants of a system. This library does not engage with the implementation of algorithms, which detect possible execution errors or other interferences. The aim of this study is to evaluate the effectiveness of TADElib in facilitating task coordination and increasing efficiency among multiple robots, especially in the Botball competition.

Throughout this paper the abbreviation TADE (= Task Administration Data Exchange) will be used to refer to the process of exchanging task administration data. It is also necessary to clarify what is exactly meant by the term task: It is used to refer to an activity conducted by a machine which has a certain output. What functions a certain task contains is specified by the user.

II. Literature Review

An important factor for TADElib is a dynamic task allocation. A dynamic task allocation should be used to increase the overall performance of a multi-robot system (MRS), which typically means to minimize the overall execution time as K. Lerman, et al. stated in their study [2].

As of today, many algorithms and tools have been proposed and developed. Moreover, there are numerous works and studies which deal with the concept of MRS and their optimization. For example, A. T. Tolmidis and L. Petrou [3] describe an auction-based algorithm for optimizing multi-objective task allocation problems, in which the robots evaluate the bids assigned to each task. Another approach, the so called multi-objective particle swarm optimisation (MOPSO) based algorithm, takes account of many instances, inter alia task execution cost and transferring time, for an efficient task allocation [4].

Furthermore, in [5] Robot Operating System (ROS) is presented as a useful tool, which enables robots to communicate with each other. The study shows that ROS can be used efficiently to create a collaborative environment for robots.

Considering the works above, there have been several revolutionary achievements in the field of MRS. This prototype of TADElib proposes a manifestation of a simplified version of the MOPSO-based approach which uses only one instance to evaluate received tasks and a novel package protocol. Nevertheless, there have not been any studies conducted regarding handling errors, which may occur during crucial tasks.

III. Project Setup

In this chapter tools which are used for the development of the TADElib prototype are introduced and explained. Relevant information and the version number for each tool are specified within the table presented in Fig. 2. Most of them are used in combination with the programming language C++.

A. Threading

A machine should be able to continue executing a task while still processing and sending TADE requests in the background. Therefore, TADElib must support threading. Threading is also used to easily control operations: One task is assigned to one thread to simply start, pause, resume, and terminate it. How it was exactly used is further described in IV-E. The authors chose the C++ library called “Threading” for the required functionalities.

B. CMake

CMake manages the compiler-independent build process of the TADElib. It produces a native build environment that assembles the executable automatically [6]. The authors used CMake to compile and build the TADElib.

C. Sockets

All members of the system are connected over a network, therefore Sockets were used to send TADE packages over the given connection. For this purpose the C++ header “sys/socket.h” is also included in TADElib.

D. Wombat

The KIPR Wombat was used for the basic robot construction shown in Fig. 1. The network card inside the Wombat enables it to host or connect to a network so that communication between robots can take place.

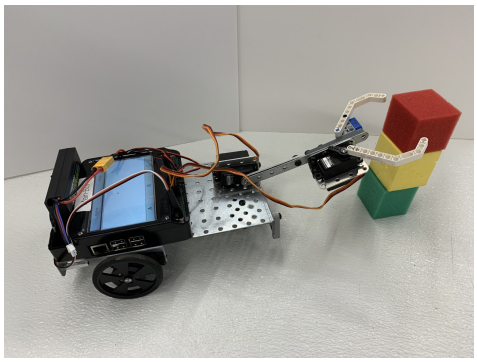


Fig. 1: A simple robot made from Botball parts and a KIPR Wombat

Project Setup

Usage	Programming Language	Concurrency	Communication	Hardware	Environment	Building the Library
Tool	C++	C++ Threading Library	Sockets	KIPR Wombat	Wombat OS	CMAKE
Version/ Specification	C++17	std::thread	sys/socket.h	-	ver. 25.6	ver. 3.0

Fig. 2: This table depicts our project setup and specifies the main tools used for the development of TADElib.

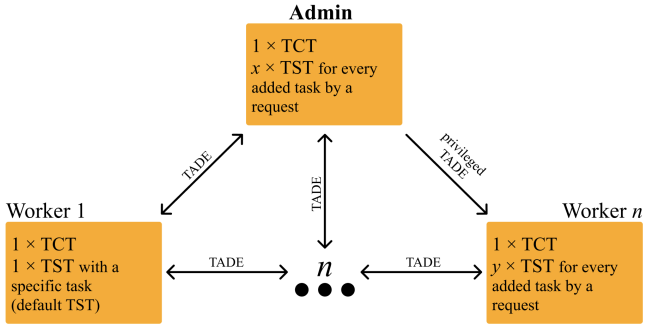


Fig. 3: A visual description of the correlation between participants. Note that the terms TST and TCT will be explained in chapter IV-E.

IV. Task Administration Data Exchange

A. Concept

As shown in Fig. 3, all participants in the system have their own jobs and the arrows indicate the correlations between them, allowing for a TADE between all devices. Within the system, there is an administrator that can be designated by the user and has the ability to send privileged TADE requests. The administrator is also responsible for calculating the total output and deciding whether a change in the plan should be made if the output falls below the expected minimum. In order for this to occur, all recorded output from worker devices must be transmitted to the administrator device, and a new plan can be established via privileged requests. An unprivileged TADE from the administrator will be treated like a regular TADE from any other device, and may be declined by the receiving participant. In the following chapter, the structure of a TADE package will be explained.

B. Structure

TADE packages have the following types, which are determined by the first digit of the TADE ID. (Note that the parameter “importance” will be explained in chapter IV-C):

- Type 0: (warning TADE) A device needs another device to pause (*P*) or even terminate (*T*) a specified task. If no tasks are specified, then all tasks, which are currently executed by the other device, are targeted.
  {*tade_id: “0xxx xxx”, importance, mode: “P”/”T“, [task_id]*}
- Type 1: (requesting TADE) A device requests another machine to perform a certain task. It can be decided whether the task needs to be executed simultaneously (*S*) with the currently executed ones, the task should be scheduled (*SC*) and conducted after the current ones are finished, current tasks should be paused (*P*) or even terminated (*T*).
  {*tade_id: “1xxx xxx”, importance, task_id, mode: “S”/”SC”/”P”/”T”*}
- Type 2: (reporting TADE) The reporting TADE is used to report output and other data to other devices (mostly to the admin).

{*tade_id: "2xxx xxx", [output], [info]*}

- Type 3: (reassuring TADE) A device reports to another one that a TADE 1 task no longer needs to be executed. Additionally, paused tasks by a type 0 can also be resumed by transmitting a reassuring TADE.
  {*tade_id: "3xxx xxx", target_tade*}
- Type 4: (responding TADE) Finally, type 4 is used to respond to another TADE, whether it was acknowledged or not. The responding TADE has the additional function to check, whether a TADE request was successfully sent to the recipient.
  If the received TADE is privileged, the response attribute of the respective responding TADE package must be *"ACK"*.
  {*tade_id: "4xxx xxx", target_tade, response: "ACK"/"NAK"*}

C. Evaluation

As previously stated, devices may accept or decline TADE content. In this chapter the evaluation of received TADE packages is described. As already mentioned, privileged TADE requests are accepted automatically and therefore do not need to be evaluated.

All tasks have a hard-coded importance attribute, which is determined by the user and should be set according to the anticipated output of the task. If a device starts a TADE type 0 or 1, which both require an importance attribute, the attribute should be equal to the importance of the task, which is supported with this TADE request. The receiving end compares the importance of the request and of its currently running tasks. A request will be declined, when its current tasks are more important.

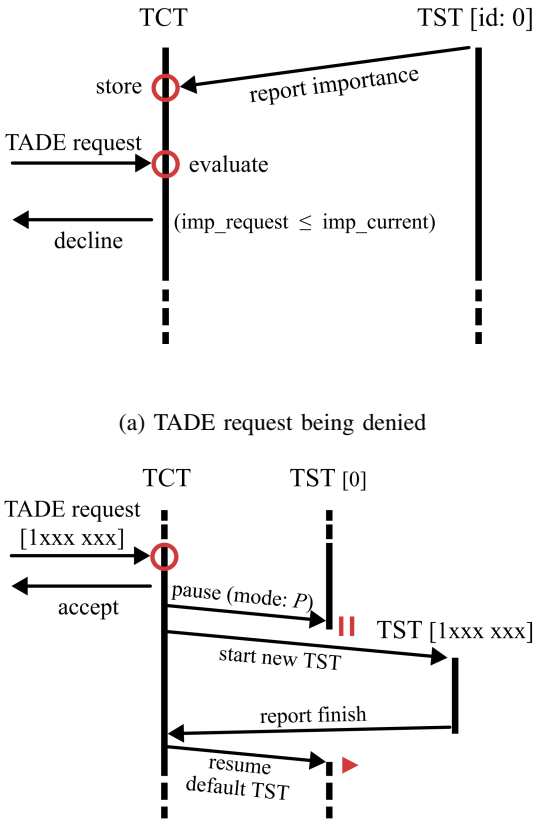
D. Connection Configuration

For a successful configuration, a connection must be established, and all system participants need to be in the same network, which is hosted by the admin. The device which is configured as admin has a file named "TADEsys.xml" stored locally. All other participants send broadcasts so that the admin can identify them. After a quick handshake between the admin and worker, the worker alongside its IP-address and some other information (if necessary) will be registered into the "TADEsys.xml" file by the admin. Similarly, worker devices have a "members.xml" file, which is a copy of the admin's " TADEsys.xml ", with the exception that the admin is included in it and marked respectively. The "members.xml" file is updated every time a new participant introduces itself to the admin. These kinds of updates are reported by a type 2 TADE to all system participants.

E. TADE Processing

There are at least two threads running on a system participant:

- TADE Control Thread (TCT) is used to receive or send TADE requests, evaluate them, update TADEsys.xml or members.xml, store or use data received from TADE type 2, etc. There is only one TCT allowed in a program.
- Task Specific Threads (TST) are threads which contain a specific task. A device has already a TST at the start of its program, which we will refer to as the default TST, and has the ability to create more of them as well as to assign tasks to them. TSTs also have an id, which is equivalent to the TADE-ID, unless the thread is the default TST. The id of such a thread is always 0.



(a) TADE request being denied

(b) TADE request being executed

Fig. 4: Diagram depicting the cooperation between TCT and TST threads

As you can see in Fig. 4a, the default TST informs the TCT about its importance value at the beginning of the program. This allows the TCT to easily compare it with TADE requests' importance attributes and decide whether a request should be rejected or executed. A detailed visualisation of a type 1 TADE request being accepted, whose mode attribute is set to "P", can be seen in Fig. 4b.

F. Cryptography

The field of cryptography involves the use of mathematical algorithms to encrypt and decrypt data, as well as to generate and verify digital signatures. The goal of cryptography is to fulfill the key security requirements, such as [7]:

- Authentication: The procedure, in which someone's identity is proven
- Confidentiality: Guarantee, that only the intended receiver has access to the data
- Integrity: Ensuring that the recipient receives the message in its original form and that the message has not been manipulated
- Non-repudiation: A method to verify that the message was truly sent by the sender

Cryptography is essential to ensure the integrity and confidentiality of TADE packages. For the duration of the testing procedure, no form of encryption between the robots was used due to simplicity. However, when TADElib is used in a real world environment, using any form of cryptography is mandatory [8]. One option for encrypting the communication between the robots is to use symmetric encryption. It works by converting plain text into cipher text using a secret key, that is shared between the robots in the current session. Symmetric encryption is a reliable and efficient way to protect TADElib from potential attacks, because it is relatively fast and requires less computational power than other methods [9].

G. Security Issues

Although the fully developed version of TADElib, which will use symmetrical encryption, is quite secure, there are still some security issues that must be addressed. One of the most significant weaknesses is the connection network between the robots. It is especially vulnerable against any type of DoS or DDoS attack [10]. These attacks paralyze the network, so that the communication is interrupted. In this case the complete communication would be blocked, and any sent TADE packages would not be processed properly. Fortunately, there are several ways to prevent these attacks: For example, by using a HTTP-based framework [11]. Another major security problem is the secret key sharing process. The secret key, which is used for the symmetric encryption, must be exchanged securely, before the communication can begin. [12]

V. Experiments

A. Using TADElib for increased Efficiency

The basic construction, which can be seen in Fig. 1, is used to conduct the experiments. In the first scenario, the yellow cube in the middle of the stack, which is also shown in Fig. 1, is needed for another task and the two other cubes need to be stacked. In the first example the robot, which is further referred to as bot A, is instructed to move the red cube away, remove the yellow cube from the stack and take the red cube to its former place and form the required stack.

In the second approach, bot A takes the red cube and moves it away, while still holding it in its claw. In order for the second robot, here referred to as bot B, to take the yellow cube away, it sends a TADE to inform bot B, that it positioned itself correctly and freed the way to avoid any collisions with it. Afterwards bot B grabs the yellow cube with its claw and can continue its previous task. After safely removing the cube, bot B sends a reassuring TADE, so that bot A can move the red cube in its prior position. By using the TADElib, all of the collisions can be avoided and the whole process is more efficient, than by using waiting periods for both robots, since this would result in an unnecessary waste of time.

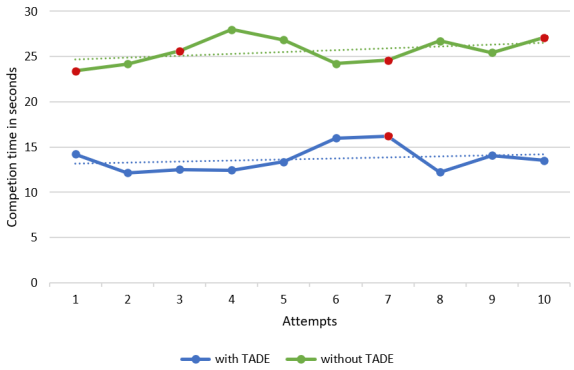


Fig. 5: This graph shows the time the bots needed to perform the task for each try in experiment A. The red points represent a failed execution in which the bot dropped a cube.

B. Using TADElib for Error Avoidance

For the second experiment, the same robot construction and setup from the first experiment was used. Only bot B had additionally a camera attached to it. In this scenario bot A drops the red cube at the top of the stack intentionally. After that, it sends a TADE to bot B, which, after receiving it, tries to detect the red cube on the ground, grabs it, and places it back on the stack.

20 runs were executed in total, and 16 of them were successful, indicating a significant reduction in the chance of task failure. In this example, the risk of failing the specified task was reduced by approximately 75%. These results demonstrate that TADElib is effective in preventing execution errors.

VI. Real World Applications

Communication between machines, such as robots, cars, and other types of devices, has the potential to revolutionize the way we live and work in the future. By enabling them to exchange information and collaborate with each other, more efficient and intelligent systems that can perform tasks and make decisions on our behalf can be created.

One of the main benefits of device communication is the ability to coordinate and optimize the performance of multiple devices at once. For example, a fleet of robots working together on a manufacturing line could communicate with each other to divide tasks, handle occurring errors, share resources, and coordinate their movements. This would allow robots to work more efficiently and effectively, improving the overall productivity of the manufacturing process.

Similarly, a network of connected cars could communicate with each other and with traffic infrastructure to optimize routes, reduce the chance of traffic congestion, and improve safety. By sharing real-time traffic data and other information,

the cars could make more informed decisions about planning the route, reducing potential accidents and improve fuel efficiency.

VII. Discussion and Conclusion

As the authors of this paper have proven through both of the experiments, the current prototype has achieved its goal to increase the overall performance of a MRS successfully. In experiment A, the average run with TADE is approximately two times faster than the average run without TADE. Execution errors were also reduced to 25%, because bot A held onto the red cube and therefore minimized the chance to drop it. While this improvement is significant, it is important to note that the TADElib is a new technology and the code provided was not fully optimized for performance. Therefore, further optimization of the code provided to the TADElib has the potential to yield even greater performance improvements for the MRS.

In experiment B, we observed that only 75% of the runs were successful. While this may suggest that the TADElib is not a reliable technology, it is important to note that the failures were not caused by the TADElib itself, but rather by the code provided to the library. Therefore, it is important to recognize that the reliability of the TADElib is highly dependent on the quality of the code provided to it.

Nonetheless, the prototype and even the TADElib itself needs some improvements as it is almost impossible to implement some applications from chapter VI with TADElib's current version.

First of all, unlike the current prototype, TADElib must use a cryptography method and the TADE network must be secured enough against network threats, especially if the TADElib is used for industrial purposes.

Moreover, an extensive system, whose participants cannot connect to the same network, cannot be configured as described in chapter IV-D. An example for such a system would be the application on autonomous cars in chapter VI. In this case, a TADE request needs to be transmitted from one participant to another until the message reaches the intended recipient.

In summary, the TADElib offers a lot of opportunities for the robotics community and Botball. Although this library implements basic algorithms developed for the concept of dynamic task allocation, which is also widely recognised in the literature, its further developed version would support many roboticists as well as the competitors of Botball in various ways.

Further development of TADElib will be continued as a project for the Botball and robotics community.

Acknowledgments

The authors of this document would like to thank Dr. Michael Stifter for his constant support throughout the work on this paper.

References

[1] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[2] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[3] A. T. Tolmidis and L. Petrou, "Multi-objective optimization for Dynamic Task Allocation in a multi-robot system," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1458–1468, 2013.

[4] F. Ramezani, J. Lu, and F. Hussain, "Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization," *Service-Oriented Computing*, pp. 237–251, 2013.

[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," *Proc. ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[6] "Overview," *CMake*. [Online]. Available: https://cmake.org/overview/. [Accessed: 27-Jan-2023].

[7] A. J. Menezes, V. O. P. C., and S. A. Vanstone, in *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 2001, p. 4.

[8] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in Industrial Networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 277–293, 2013.

[9] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, 2017, pp. 1-7, doi: 10.1109/ICEngTechnol.2017.8308215.

[10] S. A. Arunmozhi and Y. Venkataramani, "DDoS attack and defense scheme in wireless ad hoc networks," *International Journal of Network Security Its Applications*, vol. 3, no. 3, pp. 182–187, 2011.

[11] M. A. Saleh and A. Abdul Manaf, "Optimal specifications for a protective framework against HTTP-based DoS and DDoS attacks," *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, Kuala Lumpur, Malaysia, 2014, pp. 263-267, doi: 10.1109/ISBAST.2014.7013132.

[12] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography engineering: Design principles and practical applications*. Indianapolis, IN: Wiley, 2010.