CoHoN: A Fault-Tolerant Publish/Subscribe Tree-Based Middleware for Robots with Heterogeneous Communication Hardware

Steffen Planthaber and Jan Vogelgesang
DFKI GmbH - RIC
German Research Center for Artificial Intelligence
Robotics Innovation Center; Bremen, Germany
Email: {steffen.planthaber, jan.vogelgesang}@dfki.de

Eugen Nießen
University of Bremen
Bremen, Germany
Email: eugen.niessen@uni-bremen.de

***Abstract*—The increasing functionality and capability of current mobile robots is partially a result of an increased number of sensors and actors. But this larger amount of sensors and actors results in more communication between the robot's components. Nevertheless, mobile robots also have to be lightweight, they have limited size and benefit from long operation times. These constraints are limiting the choice of components, which again can result in a situation where different communication hardware has to be integrated. In general, communication in robotic systems incorporates a lot of small-sized messages of some bytes and messages of several kilobytes but very few in between. Reconfigurable robots and multi-robot systems also have some similarities to mobile ad-hoc networks as their connectivity may change during operations. CoHoN is a transparent, connection-based, publish/subscribe communication middleware for networks with heterogeneous hardware which addresses the needs of communication in robotics. It provides failure resiliance, multipath routing, quality-of-service capabilities, and very low message overhead.**

***Keywords*-Middleware; Distributed Systems; Robotics.**

I. Introduction

Communication in robotics takes place between components of a robotic system (e.g., sensors, computers, actors) and between different robots in a multi-robot system. The communication can be **heterogeneous** in the hardware used to communicate, and it can have a **changing topology**. Small messages occur often in communication between components of a robot, mostly as sensor values like joint positions, desired angles, status messages, and control commands. Thus, a low message overhead is crucial for efficiency and bandwidth. Most middleware solutions used in robotic software frameworks [1] do not take these into account.

The amount of data sent through a robotic network is often not known when constructing and building a robot. Robots in science have a long lifetime and adding additional hardware and sensors in a later stage is not uncommon, resulting in situations where the available bandwidth between the components is too low. Changing the communication normally involves reprogramming the communication routines of the software. A robotic system is composed of a large number of different electronic components, employing different communication hardwares. Each communication hardware provides a distinct set of capabilities (e.g., speed, latency, reliability etc.).

The project CoHoN is funded by the Space Agency of the German Aerospace Center (DLR), grant no. 50RA1024 and 50RA1025, with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament

Also, the network topology may change dynamically. This happens in multi-robot-systems when one robot moves out of range and also in single robots when communication hardware fails or a reconfiguration of the system occurs. A robotic network has a limited number of participating nodes, typically in the range of 100 nodes, due to space and weight restrictions.

Thus, a robotic communication middleware should operate with heterogeneous communication hardware and changing topology, but should require only a small message overhead. CoHoN is based on these requirements.

Two components of a robot are sometimes connected over more than one communication path. These cycles in the communication graph could be used for load distribution and increased failure resilience. CoHoN includes multipath routing capabilities to exploit these possibilities.

II. Related Work

Regarding communication, there are two types of robotic control frameworks. Some are using available communication middlewares, and some include custom communication abilities. In this overview only middlewares are covered that are able to communicate within a distributed network in contrast to a single device.

In the area of frameworks for robotics, supporting distributed computing an component-based infrastructure, ROS (Robot Operating System) [2] and Microsoft Robotics Developer Studio (MRDS) [3] are prominent solutions. A request and reply service is included in those two frameworks among other features. The ROS framework uses mainly publish/subscribe-based communication where messages are attributed to some topics [4]. The communication in MRDS is based on SOAP [5], which is a XML-based protocol built on top of TCP/IP. Other robotic frameworks employ existing communication middlewares [1]. OROCOS [6] and Miro [7], [8], e.g., are based on the CORBA middleware [9] while the ORCA framework [10] uses ICE [11].

CORBA and ICE, designed for large scale networks, have very low overhead compared to MRDS. But they still carry a message header size of more than 10 byte at best, hence they introduce a considerable overhead when sending small messages like sensor values. For CoHoN we want to avoid a centralized routing because this would introduce a single point of failure. Also, we have to deal with changing topology and different communication capabilities.

These requirements are solved by routing protocols developed for ad-hoc networks. An introduction and overview is given in [12] and [13].

The routing algorithm developed for CoHoN is based on directed diffusion [14] but is adapted from ad-hoc networking to the context of robotics. In [15] the directed diffusion approach is extended towards multipath routing and increased resilience against connection failures. Our routing approach is similar, but is based on a one-to-many instead of one-to-one communication paradigm.

III. Basic concepts

CoHoN is based on a topic-based publish/subscribe communication. Compared to other publish/subscribe variants, the topic-based approaches offer limited expressiveness but can be implemented very efficiently [4].

The messages sent by the publisher are called TopicItems and consist of a sequence number, used to detect duplicated and missing TopicItems, and the payload data. Each TopicItem is attributed to exactly one topic. CoHoN will integrate different kinds of communication hardwares into one network, i.e., it acts as an overlay network protocol. The TopicItems are routed by CoHoN at the interconnection points. The routing is based on a virtual circuit network, for each topic transfered between two neighbors a virtual circuit or *channel* is set up. This greatly reduces the network overhead on long-lasting subscriptions, as there is no need to transfer a unique topic identifier together with each TopicItem. The channels also allow TopicItem distribution with quality of service (QoS) by reserving the required resources during channel setup.

CoHoN uses a decentralized, request-based routing approach. Each subscription request is flooded through the network. If a node can deliver the topic, it did not forward the subscription request but returns an answer. The requesting node receives one or more answers from his neighbors and then selects where to receive the TopicItem from. This selection process continues towards the publisher.

Through this selection process a multicast tree is built up, and thus the data does not have to be sent to each subscriber separately. Although requesting a subscription through flooding results in considerable network traffic, the benefit is that this procedure discovers all possible routes and their properties in a completely distributed approach.

In robotics and many other applications, subscription requests are predominantly issued when the system is started. Later there are almost no additional subscription requests. Moreover, the subscription traffic is assigned a very low priority, thus it won't interfere with the delivery of the TopicItems. The routes may be saved to allow skipping the initial subscription request phase along with its overhead at the next startup. CoHoN also offers the possibility to disable the automatic discovery completely and to set the routes manually. There can be multiple paths between two nodes in the network. To take advantage of these cycles, additional connections can be appended to the multicast tree and used for multipath routing or as backup paths. This is similar to the braided multipath routing from [15].

In the following, the term "connection" is used for direct connection between two nodes. The term "path" is used for an indirect connection of two nodes, i.e., a path consists of one or more connections.

IV. Topic Routing

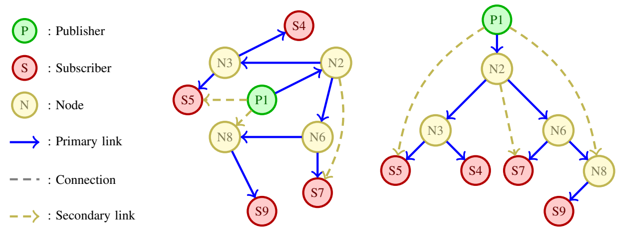
To subscribe to a given topic with unique id τ, three steps are executed. Between all connected nodes a point-to-point connection is assumed.

1) The subscription request message (SubReq) is flooded through the network. Included in the SubReq are the topic id τ and QoS constraints if required. A node receiving a SubReq and not having any current information regarding the topic forwards the SubReq to all connected nodes. It then waits for an answer.

2) A node that can deliver the topic, i.e., the publisher or a node being already part of the publication tree, sends back a subscription-acknowledge message (SubAck) over the reverse path if this path is able to comply with the given QoS constraints. Otherwise, the node returns a subscription-not-acknowledge message (SubNAck). The SubAck and SubNAck messages are forwarded over the reverse path of the SubReq message.

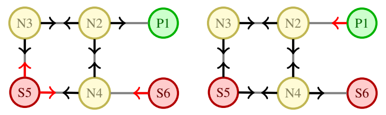
3) After receiving some SubAck messages over different connections, the subscribing node has multiple possible sources of the topic's publications. The node then selects one connection as the primary connection and reinforces it by sending a ReinforcePrimary message. Only over reinforced connections the actual publications will be forwarded. Each node receiving a primary reinforcement acts likewise: select a connection towards the publisher and send a ReinforcementPrimary. The process terminates if the publisher or a node on an already reinforced path is reached. The reinforcement of a connection has to be renewed in regular intervals, thus subscribers not anymore connected to the network will be removed from the.

For steps 1 and 2, each communication direction is treated separately. This allows to detect all available paths



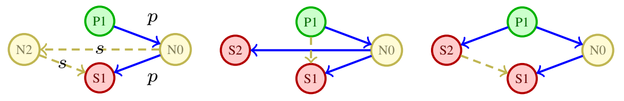
(a) Legend (b) Topolocical view (c) Topic route tree

Figure 1. Different views on the same network with one topic



(a) Step 1: SubReq (b) Step 2: SubAck

Figure 2. Messages Sent



(a) all (b) no bundling (c) no multipath, only backup path

Figure 3. Multipath possibilities

to the source even in cyclic topologies (Figure 2). Since the information gathered at step two is cached at each node, at most two SubReq and two SubAck/SubNack are exchanged over each connection within the cache lifetime. By selecting only one connection as the primary connection in step three, a tree structure is defined. This *primary tree* of τ is a spanning tree, connecting all subscribers and the publisher of τ. The publisher is the root of the tree and the TopicItems are distributed downwards in the primary tree (Figure 1).

Topic-based routing allows to use different routes for different topics of the same publisher. This is important to be able to add Quality of Service (QoS) in a later stage. Two topics of one publisher can be sent over different routes to one subscriber. Important data, like control commands, can be sent over real-time capable paths, while non real-time data, like log data, may use another path one without delaying the commands. The network traffic during subscription flooding can be decreased by using routing information already known by the nodes. Nodes, which are in the publication tree may answer directly. Additionally, nodes may cache the outcome of a request (received SubAcks or SubNacks per connection) and directly answer new requests, without further flooding. Moreover, the TopicID consists of two parts, a PublisherID and a topic number, thus routing information of other topics of the same publisher can also be used to limit the flooding of subscription requests.

To archive resilience against broken connection additional connections might be attached to the primary tree. These backup connections, called secondary connections, are reinforced by ReinforceSecondary messages. The secondary reinforcements continues towards the publisher until the primary tree, another secondary connection or the publisher is reached. Each node is aware of its depth in the primary tree from the subscription phase, this information is used to ensure the correct direction of the secondary path.

Secondary connections might be used in different ways:

As backup path: The connection is not used until the primary connection breaks. The breakdown is detected either by the driver layer or because a periodic transmission has been missing for some time. However, the latter is only possible for periodic transmissions, which are quite common in the sensor domain. When a breakdown is detected, the node selects one of the secondary connections as the new primary connection. The selection is communicated by sending a ReinforcePrimary message.

As auxiliary path: The secondary connection may help to detect lost topic items. For this, so-called TopicStubs are send via the secondary connections. A TopicStub contains only the most recent sequence number the sending node has seen. The node receiving the TopicStub is then able to detect lost topic items and either asks for a retransmission or selects the secondary connection as new primary connection. The use as auxiliary connection is especially suited for aperiodic and large messages, e.g., environment maps.

As alternative path: The secondary connection may be used as alternative connection if the primary connection is congested. This is also known as *multipath routing*.

V. Multipath Routing

When CoHoN determined multiple path possibilities, multipath routing may be available given certain circumstances. Let the secondary path s and the primary path p converge at node N0, i.e., the reinforcement of s stops at N0 (see Figure 3(a)). Let S1 be the node in p just below N0. If s heads into the tree rooted at S1, then s could be used used as an alternative to the connection from N0 to S1. If the connection does not head into the same subtree, multipath is not possible (Figure 3(c)) The prerequisites just given are checked by a *cycle probe*. The cycle probe is a special message, which starts at N0, travels down into the tree over s and then back to N0 over p. Only if the given prerequisites are fulfilled, the cycle probe will reach again N0.

A TopicItem received via an alternative route is forwarded down the primary tree (as usual) and also up towards the parent node in the primary tree. The TopicItem is sent upwards the primary tree as long as needed, i.e., till all subscribers in the tree below S1 could be reached (Figure 3(b)).

If there are no other subscribers or multiple primary outgoing links within this cycle, both routes (primary and secondary) can be bundled. This means the sending node may send the topics in an alternating way via the one or the other interface. This bundling would be possible in Figure 3(a), but not in Figure 3(b). Mulitpath routing helps to distribute bandwidth utilization, but may result in an disordered delivery of the TopicItems. The subscriber has to reorder the messages using the included sequnce numbers.

VI. Future Work

Currently, the project is within its implementation phase and thus there are no experimental results available. A first implementation of the routing algorithm in a network simulator showed the general feasibility of the approach.

Further development will include the Quality of Service (QoS) functionalities. The possibilities provided will be strongly dependent on the communication hardware used. Some QoS features are in particular useful in the field of robotics: latency and jitter. A minimum latency must be supported for control commands, e.g., security shutdowns. A minimum jitter, which is a maximum deviation of the latency value, must be applied for sensor values, which are used in motor or joint controllers.

The implementation will be followed by an valuation of the re-routing capabilities and the resulting fault-tolerance of CoHoN. To evaluate CoHoN, connections will be disturbed or completely disconnected in order to test the re-routing and the detection of failures. The hardware test setup will include Ethernet, CAN Bus, PROFIBUS, and SpaceWire. Other common communication methods, like RS232, will be added later.

The selection of hardware covers most of the communication principles used in robotics, i.e., Single Master Bus (PROFIBUS), Multi-Master Bus (CAN), Point to Point (SpaceWire), and Ethernet (Point to Point on driver interface level). The straightforwardness of the underlying routing (virtual connections with channels) was also chosen in order to be able to implement CoHoN nodes on micro-controllers or FPGAs in a later stage of development. These have low processing power but are used frequently by sensors and actors.

VII. Outlook

We have presented the design prinicipal and routing approach of CoHoN. CoHoN is not meant to replace existing robotic frameworks but rather to provide an alternative for exchanging their data between distributed nodes. It does not include data marshaling but is designed to submit blocks of data, thus CoHoN can be adapted easily to existing communication methods. The chosen publish/subscribe approach is used by many frameworks (e.g. ROS) and has already proven to be flexible and powerful.

CoHoN will allow an additional topic naming based on textual tags. In a Service Discovery process, the tags are resolved to actual topics. This allows to support different naming schemes used by robotic frameworks.

References

[1] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, “Middleware for robotics: A survey,” in *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*. IEEE, 2008, pp. 736–742.

[2] 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,” in *International Conference on Robotics and Automation*, 2009.

[3] J. Jackson, “Microsoft robotics studio: A technical introduction,” *Robotics & Automation Magazine, IEEE*, vol. 14, no. 4, pp. 82–87, 2007.

[4] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of Publish/Subscribe,” *ACM Computing Surveys*, vol. 35, pp. 114–131, 2003.

[5] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, “Unraveling the web services web: an introduction to soap, wsdl, and uddi,” *Internet Computing, IEEE*, vol. 6, no. 2, pp. 86–93, 2002.

[6] H. Bruyninckx, “Open robot control software: the OROCOS project,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2005, pp. 2523–2528.

[7] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, “Miro-middleware for mobile robot applications,” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 4, pp. 493–497, 2002.

[8] G. Kraetzschmar, H. Utz, S. Sablatnög, S. Enderle, and G. Palm, “MiroMiddleware for Cooperative Robotics,” *RoboCup 2001: Robot Soccer World Cup V*, pp. 95–110, 2002.

[9] T. H. Harrison, D. L. Levine, and D. C. Schmidt, *The design and performance of a real-time CORBA event service*, ser. OOPSLA ’97. New York, USA: ACM Press, 1997.

[10] A. Makarenko, A. Brooks, and T. Kaupp, “Orca: Components for robotics,” in *International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 163–168.

[11] M. Henning, “A new approach to object-oriented middleware,” *Internet Computing, IEEE*, vol. 8, no. 1, pp. 66 – 75, 2004.

[12] J. N. Al-karaki and A. E. Kamal, “Routing Techniques in Wireless Sensor Networks: A Survey,” *IEEE Wireless Communications*, vol. 11, pp. 6–28, 2004.

[13] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, pp. 325–349, 2005.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.

[15] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly-resilient, energy-efficient multipath routing in wireless sensor networks,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, p. 11, Oct. 2001.