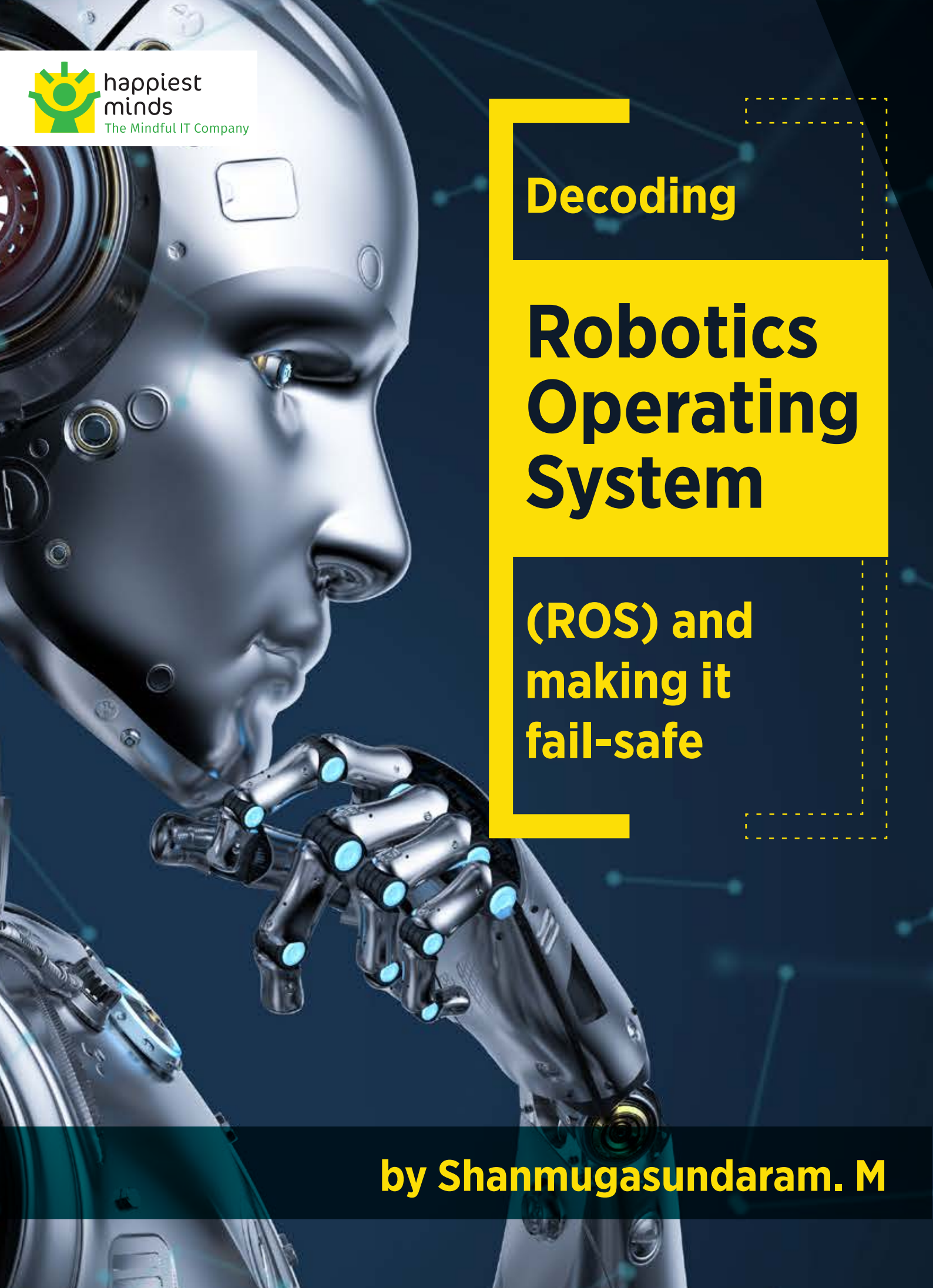


happiest minds
The Mindful IT Company

Decoding Robotics Operating System (ROS) and making it fail-safe

by Shanmugasundaram. M

Introduction

Robotics, the interdisciplinary branch of Engineering and Science strongly backed by mechanical, electronics and computing technologies, is already known for its “smartness”. In 2015, Gartner had predicted that by the year 2025, software, robots and other smart machines would take over one in three jobs currently performed by humans. However, until recently, traditional robots were programmed only for deterministic environments and were deployed in highly predictable environments where they were not challenged enough with surprises. Robot manufacturers designed and built robots with their own proprietary operating and programming systems. These systems could not be leveraged by other robot manufacturers due to proprietary communication and allied systems that lacked transparency.

Fortunately, with time, hardware became less expensive and processing power became affordable and ubiquitous. Digital minds began pondering over the idea of developing “intelligent” robots—autonomous robots that could learn by themselves and act without human intervention. Smart robots would be ones that could take advantage of artificial intelligence (AI), building capabilities by learning from their environment and their own experience. They would be able to collaborate with humans, working them and learning from their behavior. Was it an achievable goal, and if yes, how easy would the journey be?

To accomplish such an autonomous scenario, it was important for people from disparate backgrounds (such as artificial intelligence, electronics, distributed computing and communication technologies) to work together. Secondly, a common platform that was compatible with different robots was required, and an ideal scenario would be one where this platform was open source. But, was such a robotic platform available? Yes, and Robotic Operating System (ROS) provided the answer.

A Robotic Operating System that is not an operating system!

A traditional operating system runs a computer, controlling the various applications and resources running on it. Operating systems facilitate the completion of different actions—sending a message, opening a file, etc. Hence, it would be misleading to refer to ROS as an operating system. If we apply the analogy of traditional operating systems to ROS, an ROS provides certain services or a set of libraries and tools which any robotic application can utilize to fulfil a set of requirements. It is in fact a meta operating system that operates on top of a traditional operating system, usually Ubuntu.

Ever since its development, ROS has been running on Ubuntu, and hence, it is considered the official operating system of ROS. However, there have been various instances of successful deployment of ROS in other operating systems such as Arch Linux, albeit through container systems. An interesting development in the area has been initiated by Microsoft where it has started porting ROS onto Windows, calling it ROS 1.

ROS is a pub-sub-based framework with all communication taking place through messages as topics.

So, what is ROS made of?

At the basic level, ROS is a framework with publish and subscribe mechanisms at its heart. ROS integrates many other libraries and frameworks into it and this alleviates the need to install separate libraries. As an example OpenCV, which is a separate image processing library, is bundled with ROS. This is a convenient option for developers as they can use OpenCV directly within ROS without worrying about unforeseen dependencies. Image processing is such an integral requirement for robotics that the bundling of OpenCV with ROS offers obvious benefits.

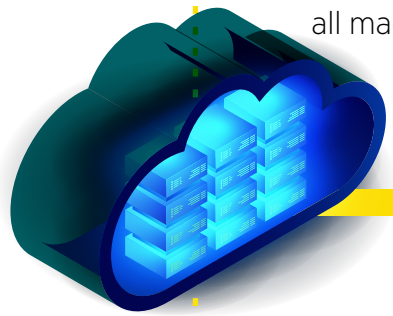
Nodes are created based on specific needs and these nodes talk to ROS Core (the central router) appropriately.

ROS is a pub-sub-based framework with all communication taking place through messages as topics. Nodes are created based on specific needs and these nodes talk to ROS Core (the central router) appropriately. The remaining nodes are referred to as ROS nodes. The key functionality of ROS Core is to handle messages from all nodes connected to it and to pass on these messages to other nodes based on subscription of specific nodes to specific topic/s.

Distributed computing in ROS

The inherent distributed architecture of ROS makes it a powerful framework for robotics. In most cases, the robot might not have a powerful in-built computer, and it may not be possible to run some computing-intensive applications (image processing, image recognition, neural networks, etc.) on the robot itself due to high computing and power expense. Fortunately, this challenge can be addressed with distributed computing where some other computing entity does the hard work.

Another challenge that the distributed architecture poses revolves around seamless communication and data integrity. However, ROS offers a solution here as well by allowing ROS nodes to run in a different physical computer. This enables hard computational tasks to be shifted to nodes in powerful computers from the robot. The robot does what it can do best and the rest is completed in the Cloud. All tasks are fulfilled inside the ROS framework. All ROS nodes in the local network communicate with each other based on topics, and hence, computations are bound to be seamless irrespective of the location where the nodes run—in the computers inside the robot, or physically outside the robot. The only caveat here is that all machines need to be in the same local network.



ROS offers a solution here as well by allowing ROS nodes to run in a different physical computer.

Is ROS fail-safe?

Robots cannot fail just like that, and by its inherent nature, robotics needs to be fail-safe. But, is ROS really fail-safe, and if the answer is “no”, what can we do to make it fail-safe? ROS is a set of tools that makes robotics successful, and these tools also work to make the mechanisms fail-safe. It is important to carefully monitor ROS Core, the central master for coordinating communication with other nodes, for any failure. This can be done via a reporting node connected to ROS Core and sending/receiving sample monitoring messages to verify and ensure that ROS Core is running successfully. ROS nodes will connect to ROS Core when they begin functioning, and hence, it is important for ROS Core to start running before any ROS nodes are started.

ROS is a set of tools that makes robotics successful, and these tools also work to make the mechanisms fail-safe.

If ROS Core fails in a running system, then the ROS nodes which were connected to ROS Core will not try to re-establish the connection even after restarting ROS Core. Hence, it is important to monitor the whole system to ensure ROS Core is up and running. Each node has to have a strong mechanism to gracefully restart the connection if ROS Core fails. If a node fails to connect, an in-built mechanism should be able to restart the nodes to make the entire system fail-safe. As a rule, it is imperative to manage the ROS in an intelligent manner to bring the system back from failure.

Useful tools within ROS

Here, we discuss commonly used tools in ROS that make lives of developers a lot easier. Out of the many readily available tools that make application development quick and easy, “Catkin”, a compilation tool, is quite popular. Though makefiles help streamline the compiling process, they are usually complicated and most developers consider them a chore. Catkin goes a step further and makes makefiles friendlier and approachable. The compiling process takes care of all the dependencies and installs without any complication.

CATKIN

There are other command line tools and graphical tools that enable developers and users to monitor the system easily. The command “rosnode list” is worth a mention due to its resourcefulness. For a node to be inspected, the command “rosnode info nodename” has to be used. The command lists topics that the node is subscribed to. It also lists topics it publishes and the services it offers.

ROSNODE LIST

Another interesting graphical tool is the rqt_graph which displays all nodes in a graphical format along with topics and publisher-subscriber relationships.
Finally, any node can be killed with Ctrl-C or “rosnode kill nodename”.

RQT_GRAPH

Conclusion

By its very architecture, ROS is distributed and modular making it suitable for the robotics arena. It is also one of the most popular open-source robotics frameworks available today supported by an always-ready-to-help community. Interestingly, new packages are constantly being added to ROS, making it more convenient and useful than ever before. To top all, with major robotics as well as non-robotics companies embracing ROS, and adopting ROS into any kind of robotics work-either academic or commercial-it is likely to create a win-win situation for everyone involved.

About the Author:

Shanmugasundaram. M (Shan) is a prolific inventor and creator of several products. He owns 14 patents in the areas of telecom, automotive (OBD II, J1939 etc.), Machine to Machine (M2M), etc. Shan possesses 16 years of IT industry experience in R&D (telecom, automotive, M2M, etc.), filing patents, inventions, converting inventions into successful products, customer projects and maintenance projects.

An accomplished master and successful implementer in M2M technologies, Shan has proven himself time and again in creating and deploying real-world cutting-edge M2M products such as Logica EMO, Static Asset Monitoring and Retail Innovations.

An accomplished master and successful implementer in M2M technologies, Shan has proven himself time and again in creating and deploying real-world cutting-edge M2M products such as Logica EMO, Static Asset Monitoring and Retail Innovations.

These inventions have won him numerous awards, including **Golden Peacock Award, a NASSCOM award and The Economist award.** The inventions also provided Logica with the much-needed global exposure in different domains, including automotive technologies.

Shan and his inventions have been covered in leading national and international publications, including Times of India (http://bit.ly/ROS43M), Hindustan Times, The Economist (http://bloom.bg/Uvg74g), Electronics for You, and The Livemint (http://bit.ly/VHsW02).