

Unmanned Georgia Tech Helicopter Flies with NDDS

An Application Note by Real-Time Innovations, Inc.



"We found that a good mix of the right software tools and middleware cut development time and minimized confusion."

"Without NDDS, our goal could never have been accomplished."

Aaron Kahn, chief designer of the Georgia Tech helicopter.

Abstract

The schedule is ambitious. The deadlines can't be missed. Not a line of code has been written. There's no time for prototyping. There aren't enough developers to do the work.

Sound familiar? Engineers at Georgia Institute of Technology's Unmanned Aerial Vehicle Research Facility (UAVRF) were recently faced with just such a situation. They were developing a small, autonomous, radio-controlled helicopter to fly in the International Aerial Robotics Competition (IARC) and they had only a short time to put it all together. One of the critical components was the software that their unmanned helicopter would use for communications with their ground station.

The communications software had to be small, simple, intuitive to use, and available off the shelf. They found what they were looking for in NDDS from Real-Time Innovations (RTI). NDDS eliminated time-consuming and complex network programming. Using NDDS, the Georgia Tech team was able to get its helicopter flying under complete computer control in time for the competition.

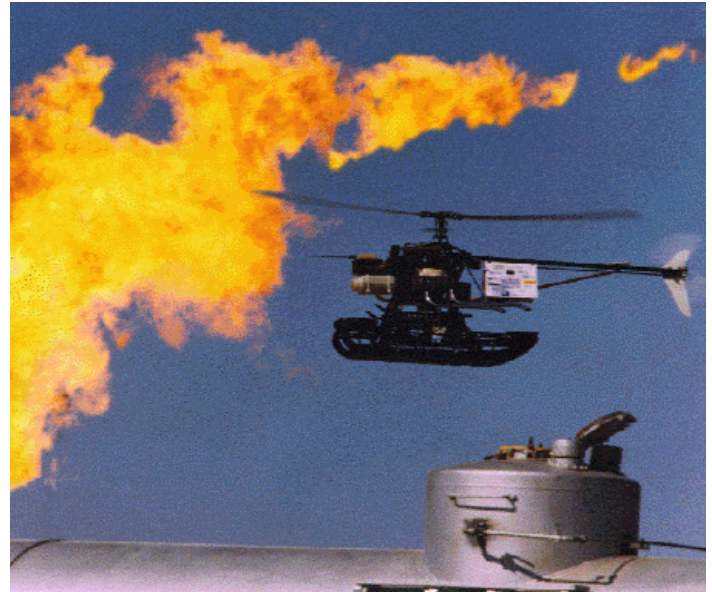


Figure 1. A Hazard From The International Aerial Robotics Competition

The Ultimate Collegiate Challenge

The IARC is sponsored by the Association for Unmanned Vehicle Systems (AUVS) International. It captures the competitive spirit of graduate engineering students from many of the top universities around the world. The AUVS sponsors annual autonomous robotics competitions in aerial, ground, and undersea categories. These competitions require undergraduate and graduate students to design and build unmanned vehicles that attempt to complete a predefined mission. The AUVS competitions are billed as "The Ultimate Collegiate Challenge."

An IARC mission is often very complex. For example, this past year's competition required the unmanned aerial vehicles (UAVs) to complete a search-and-rescue mission over a five-acre area that simulated an urban natural disaster. This simulated disaster area included hazards such as natural gas leaks and broken water mains - where actual flames and waterspouts were shooting into the air. (See Figure 1). For the mission, the UAV was required to locate simulated survivors of the disaster and identify various hazardous materials strewn throughout the area.

The Georgia Tech Entry

Preparation for such a mission can take years. The problem for the graduate students at the Georgia Tech UAVRF was that they did not have years; they had only five months to put together a working helicopter from scratch. They had to select the hardware components, build the helicopter, write and test the software, and fine-tune the flight controls. Even though autonomous aerial flight was not new to the Georgia Tech team, the task was still daunting.

First, the team selected the various hardware components. In addition to normal helicopter equipment, the UAV was equipped with a small circuit board containing an AMD 486DX5-133 processor, 128 MB of RAM, and 64 MB of flash memory. It also was given a Global Positioning System, a SONAR altimeter, and an Attitude and Heading Reference System.

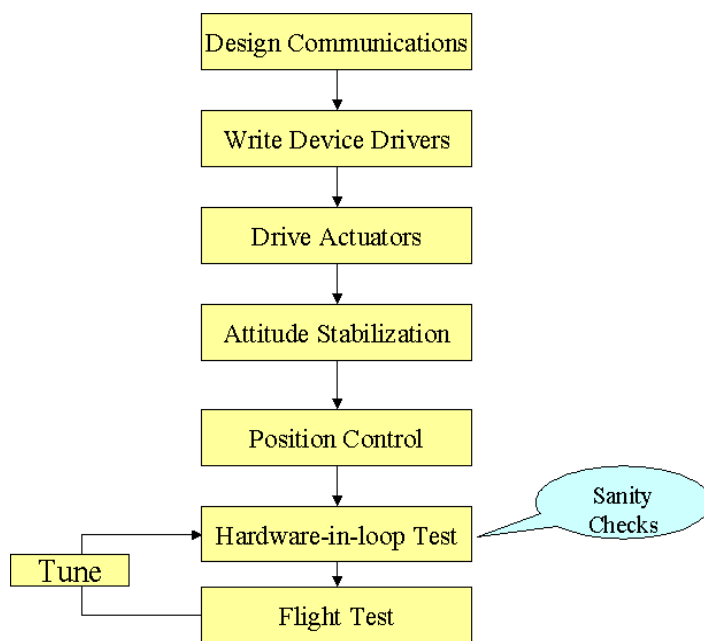


Figure 2. The Software Development Process.

Communications between the helicopter and the ground station were provided over wireless Ethernet. While some members of the team worked on integrating the various hardware components, others selected the software components used on the project. RT-Linux was chosen as the operating system for the embedded processor on the helicopter, and Red Hat Linux was selected as the operating system for the ground station computer. To minimize software development time, the team knew that they

needed to use as many off-the-shelf software components as possible. "We found that a good mix of the right software tools and middleware cut development time and minimized confusion," said Aaron Kahn, chief designer of the Georgia Tech helicopter.

The first step of the software development process was to design a communications subsystem. (See Figure 2.) This would ensure that all the various software systems could communicate in an easy, scalable, and flexible way—regardless of whether the systems were local or remote (that is, across the wireless network).

The team decided to use off-the-shelf communications middleware to save both time and resources. "Using off-the-shelf middleware helped us to focus more completely on our overall objective rather than on the details of message-passing algorithms or network programming idiosyncrasies," said Kahn.

NDDS: Off the Shelf and Into the Air

Several off-the-shelf options for middleware were considered. "We looked at CORBA, but not only was it too large for our embedded application," said Suresh Kannan, chief software architect for the project, "but the ramp-up time to use it was prohibitive. Just to send a few data values from one system to another required too many rules, conventions, and standards for us to figure it out expediently." In contrast, NDDS—the real-time, publish-subscribe middleware that was selected—required virtually no ramp-up time. "What is really impressive about NDDS is that it is extremely robust, yet both simple to use and very fast."

The publish-subscribe model for messaging is shown in Figure 3. Nodes that produce information (publishers) create "topics" (such as temperature, location, or pressure) and publish "issues" of that topic. The issues are then delivered to all subscribers that have declared an interest in the topic. The middleware handles all the transfer chores: message addressing, data marshaling and demarshaling (so subscribers can be on different platforms than the publisher), delivery, and flow control.

For example, in the case of the Georgia Tech helicopter, differential corrections were labeled with the topic `gs.diff` and published to the middleware. Any other system, either local or on the network, simply subscribed to the `gs.diff` topic and received new issues automatically as they were published. Both intertask communication and ground-station-to-helicopter telemetry systems utilized NDDS middleware.

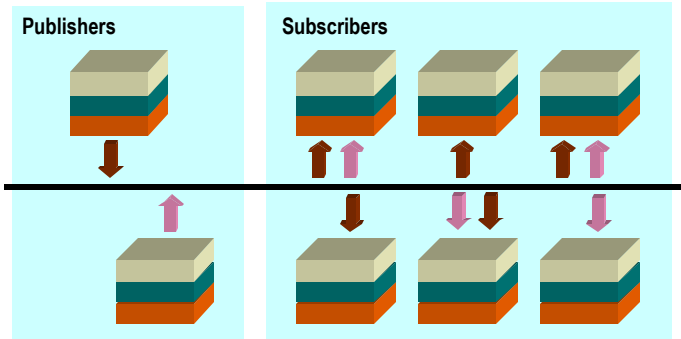


Figure 3. Publish-Subscribe Middleware.

Publish-subscribe middleware simplifies many-to-many communications. Publishers just declare a publication, define the topic and give the issues to the middleware. Subscribers just declare interest in a topic. The middleware takes care of all the network addressing and sends the issues only to the nodes that declared an interest.

Once the team established messaging between software subsystems, it "never had to bother with it again." It saved the project weeks, or perhaps even months, of development time. "Without NDDS, I know that we would have spent a lot of our time trying to hack together some type of communications code which may not have ever worked," said Kahn. "Without NDDS, our goal could never have been accomplished."

Although a lot of data was sent to and from the aircraft over the wireless network, NDDS overhead was never a concern. In fact, NDDS overhead remained negligible while handling data at a rate that, in one test, neared the physical limits of the wireless hardware.



Figure 4. The Georgia Tech Helicopter at the Competition

Conclusion

By utilizing off-the-shelf publish-subscribe middleware, the Georgia Tech team was able to meet a very aggressive software development schedule. There was still plenty of work to be done when the team arrived at the 2000 International Aerial Robotics Competition, but debugging communications software was not a part of it. As might be expected, a number of aircraft in the competition crashed during the test flights that preceded the final competition. However, the Georgia Tech UAV was one of only two crafts to fly completely under computer control and make it through to the final stage of the competition. As a result of this success, the Georgia Tech UAV Research Facility is using NDDS on new research projects--and, of course, any future aerial robotics competitions.

Additional Information

NDDS-Network Programming Made Simple

NDDS is network middleware that simplifies the development of distributed, real-time applications. Its publish-subscribe architecture is the natural, straightforward way to remove the complexity from many-to-many communications. Its sophisticated delivery system distributes time-critical data quickly, efficiently, and deterministically.

NDDS provides:

- Field proven publish-subscribe communications
- Simple, easy-to-learn API
- Fast, deterministic data delivery
- Communications over a standard IP network
- Comprehensive services for monitoring and control

NDDS simplifies the design, development, and deployment of distributed real-time applications.

NDDS is the communications engine for the WaveWorks product family of middleware and tools for real-time networking. WaveWorks is designed specifically to help developers through the full develop, debug, and deploy process.

For more information visit Real-Time Innovations, Inc. at www.rti.com.

About RTI

Real-Time Innovations, Inc. (RTI) is a leading developer of new tools and architectures for the growing real-time software market. The company's products help real-time developers analyze and understand embedded systems, speed development of distributed real-time systems, and coordinate the work of teams of programmers developing large projects. RTI products power applications ranging from network monitoring to the Space Shuttle launch-control system. RTI has thousands of customers, including leading companies in aerospace, semiconductor equipment, telecommunications, robotics, and industrial automation. Together with our customers, we are *Shaping the Future of Real Time™*.

Contact Information

Real-Time Innovations, Inc.
155A Moffett Park Drive
Sunnyvale, California 94089

Web: www.rti.com
E-mail: info@rti.com

Phone: +1 (408) 734-4200
Fax: +1 (408) 734-5009

In Europe:
Phone: +49 (089) 74120 106
Fax: +49 (089) 79120 102

