

# Tornado: Real-Time's Connection to the Network

**Network development makes it possible for a engineers to acces the development target with only one link to it. But when working with resource limiteded targets not supporting full networked development, it can get hard to a team to provide runtime maintenance of embedded applications. Tornado development environment from Wind River Systems provides a solution by using the advantages of network development without loading the target with a full network stack.**

**T**raditionally, embedded systems have been islands of automation: self-reliant machines that contain everything the system needs to work over its lifetime. Although it might be possible to make changes, they would typically have to be made by a field service engineer supplying the updates using a local terminal or with a ROM change.

The dramatic growth in Internet usage and the applications that service that demand have indicated how networked systems can be exploited for advantage on desktop systems. At the same time, the Internet and other network technologies are making it possible for embedded systems developers to streamline the way in which they create applications and maintain them over their full life cycle. Network technologies are also helping to create new types of embedded system, such as the smart portable phone and the personal digital assistant (PDA), as well as boosting the market for infrastructure products, such as network switches and routers.

## NETWORKED DEVELOPMENT

Time-to-market constraints have emphasised the importance of making the development target available to as many engineers as possible. Networked development makes this possible by linking everybody on the team through one cable to the target. This makes full use of the multitasking nature of an RTOS environment as each engineer will be working on a different task, avoiding potential conflicts.

## WORKING WITH RESOURCE LIMITATIONS

Not all targets have the resources or memory available to support full networked development directly.

With the developers of deeply embedded systems turning to the advantages of RTOS-based systems design, it is not cost effective to fit them with networking support simply for development. The Tornado development environment from Wind River Systems was designed to address this potential problem by

**A key feature of the target server is its ability to incrementally load object modules into the target system.**

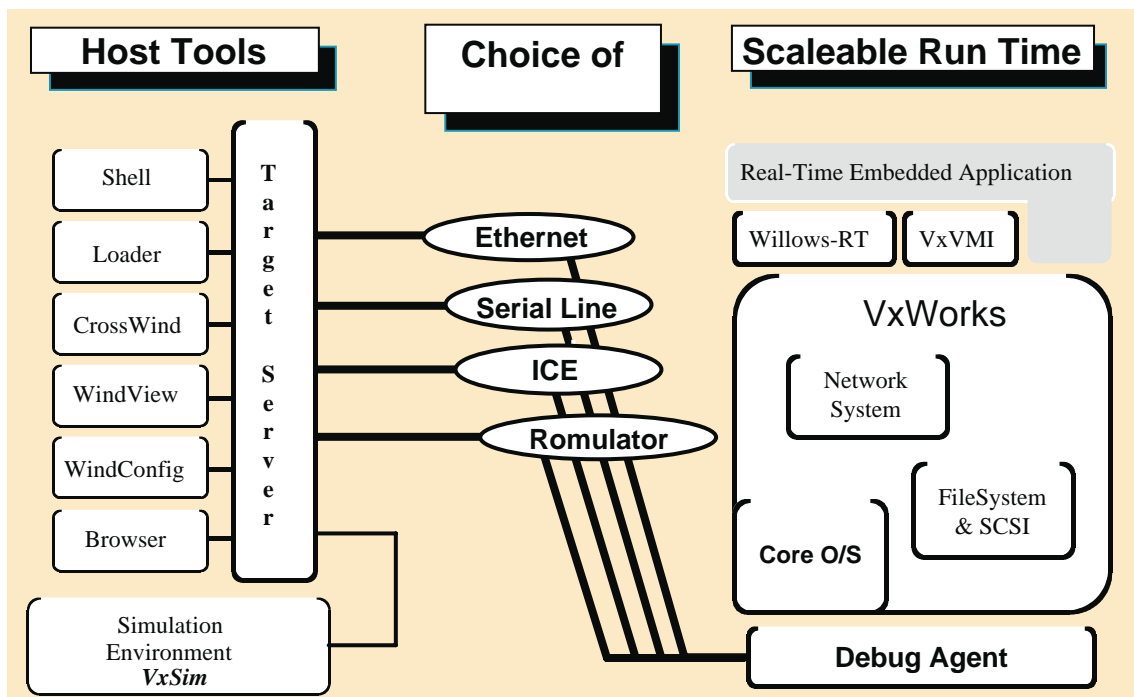


Figure 1. Tornado provides a variety of debug tools and target connection items

# TORNADO

making it possible to provide the advantages of networked development without loading the target with a full protocol stack.

Its client-server design lets multiple tools interact with the target and only calls for a minimal amount of resource on the target itself. Unlike other approaches to networked development, Tornado puts most of the debug and networking intelligence into a target server, running on a PC or workstation, which is generally attached to the target through a serial port or a custom connection.

The connection does not have to be to the target itself. For debug support with very little intrusion, Tornado can communicate with a variety of hardware-assisted debuggers such as incircuit emulators, BDM, JTAG or ROM-based emulation tools. That makes it possible to make use of the advantages of RTOS-based development even on systems with severe memory and I/O constraints.

Besides providing a number of services traditionally located on the target such as a debug monitor, the target server acts as a broker for a number of client tools which may run locally or across a network. Tornado uses a standardized interface for communication with all of the client tools, providing a set of simple operations that are independent of the specific tool types. However, the primitives needed to support debuggers and other tools are all supported.

For example, a debugger needs the ability to read and write memory, create execution contexts, examine the symbol tables of loaded object modules and read or write registers. Tornado provides the support for these functions and ensures that tools do not conflict with each other if they try to access shared resources.

## TARGET SERVER SERVICES

A key feature of the target server is its ability to incre-

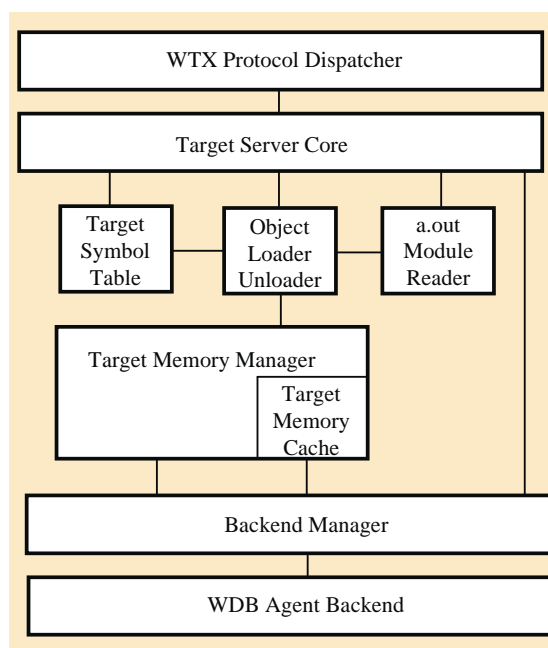


Figure 2. Internal architecture of the target server

mentally load object modules into the target system. The target server manages this process by using the symbol table contained in every object module to build a system-wide table of loaded functions, variable names and module identifiers. Names from system and application modules alike can be added to the system symbol table. Because modules can be added dynamically, development cycles are much shorter: when a developer changes a particular feature, only the affected module is recompiled and reloaded, not the entire application.

This symbol table is the heart of many of the most significant development aids. First, the loader itself uses the system symbol table to resolve undefined references in modules being loaded, dynamically linking newly loaded modules to previously loaded modules. The system symbol table is also used to provide an interpretive shell with access to all system and application modules that have been loaded.

Runtime linking makes it easy to have genuinely shared subroutine libraries, in which a single copy of a set of subroutines can be used by several tasks, rather than requiring each task to be linked with several copies of needed subroutines. As a result, there is no inherent distinction between the runtime system modules and application modules. As a result, all runtime facilities are easy to access, modify and extend.

In addition, the target server provides the ability to unload modules from the system. When a module is removed, all the associated symbols are removed from the symbol table and its resources, such as memory, are reclaimed. As object modules can come in many formats, the target server isolates the object module format reader from the core of the target server. This makes it easier to support new formats as they appear independent of version updates of the target server.

The dynamic nature of Tornado makes it very easy to update systems, even when they are out in the field. As problems arise, new tasks can be added to take care of them or existing tasks and modules can be replaced with new versions. A system with limited resources may need to be connected to a local target through an in-circuit emulator or a serial debug port which may mean field engineers having to pay a visit to the site.

## DEVELOPMENT IN THE FIELD

For systems that have the headroom to support a network connection directly, the convenience of networked development with dynamic control over the target can be extended to remote debugging over long distances: the network does not have to be a local area network (LAN). The Clementine space probe used to map the surface of the Moon employed an onboard computer running under VxWorks, the operating system portion of Tornado. When it developed a problem with the tracking mechanism, the engineers were able to upload a task over the radio link and prevented the fault from wrecking the entire mission.

Because the Tornado tools were designed for use

over a network, they can easily be used remotely which means that engineers have access to the same tools that they would employ for initial development. For example, if problems arise in the field that are due to task interaction or timing problems and not program bugs, tools such as StethoScope or WindView can be used to visualise the system. With the support of those advanced tools, problems can be resolved quickly and often without a visit by the engineer in person.

The ability to dynamically load and unload tasks remotely and to maintain run control over the target has been in VxWorks and Tornado for 10 years. Many customers have used it as a transparent way of updating applications. Now, through Java and related developments, these functions can be made available to users and not just engineers. If a network connection is already in place on the target, it can be reused to provide a more advanced user interface without increasing the end cost of the system.

For example, a multifunction network printer needs its network connection to accept printing jobs, but the network connection can also be employed for control. Usually, control over the printer is achieved through a limited number of keys with feedback through a small LCD. Using a Web browser, a user on the network can get information from the printer such as the amount of paper that it has left or where paper has jammed.

Using downloadable Java applets, the user can exercise control over more advanced functions. If a colour printer, an applet could be used to let the user have more precise control over the colour balance by providing an onscreen preview of the print job.

One advantage of a multifunction printer is that it combines the functions of a number of office peripherals, such as copiers, fax machines and page printers. With the help of Java applets, those functions can be combined in more innovative ways. For example, a document sent to the printer might also be faxed with a standard cover sheet in the same operation.

Control over the job would be handled by an applet which may also take advantage of other systems on the network. If there is a fax server, it might be used to provide lowest-cost routing over the company's network in place of the printer's direct PSTN connection for some of the faxes. By putting this intelligence into an applet, the rules for fax routing can be changed easily. As requirements change, new applets can be downloaded or activated to service them.

Downloadable tasks and objects can be used to extend the lifetime of a printer. For example, as new image formats are defined, changes to the printer software could be downloaded from the Internet and sent on to the printer, streamlining its operation without troublesome ROM changes.

By adding a relatively low-cost network interface, other types of hardware can benefit. For example, a PBX is typically programmed through a dedicated terminal by an engineer and through a handset by the

user. Because telephone handsets have a only limited numeric keypad and perhaps a small display even changing simple parameters, such as the time of day, can be excessively time consuming.

Using Internet technology, it is possible to give the PBX a user interface that appears on any PC, making its features much more accessible. As the network connection can be used to support voice-mail and other computer-telephony integration (CTI) features, the networked user interface can be used to support the PBX features that are rarely used if they can only be controlled from a handset. It is much easier to control a conference call from the display of a PC with point-and-click menus than from a numeric keypad.

## ADDING NEW SERVICES

If new services are added at other points on the network, embedded systems designers can ensure that their products can make full use of them if they are appropriate, and derive more revenue without necessarily having to deploy new hardware. By downloading tasks or applets that can tie these services into the existing setup, the system can offer them in a more cohesive way to the users.

For example, in the case of a PBX, the addition of a router elsewhere on the network which has support for sending real-time voice traffic over the Intranet or Internet could be used to cut the overall telecom bill. With the addition of a new task to control the routing, calls could take advantage of the lower cost of the packet-switched Internet instead of a direct circuit-switched connection. An associated applet would let the network administrator set up the rules for which calls would be routed and at which times.

## THE NETWORK CREATES NEW PRODUCTS

Some applications for embedded systems are now appearing thanks to network technologies and their requirements are having an effect on the additional services that an RTOS provides. For example, GSM's support for fax and data networking makes it possible to build mobile handsets that add the functions associated with those of a personal digital assistant (PDA). With a larger display and more extensive keypad, it is possible to let the user send and receive e-mails and even look at Web pages.

This type of system demands more from the operating system. As well as the protocol support, it needs more advanced user interface and graphics support. Although a desktop-oriented operating system has those features, it generally demands too much memory to be cost-effective.

The support tends to be monolithic: everything is included simply so that the system can boot successfully. Also, without hard real-time support, it is very difficult to design the user interface portion of the application so that it does not interfere with the timing of vital communications systems such as the GSM protocol processing. Although it is possible to use multiple processors to divide the workload between real-

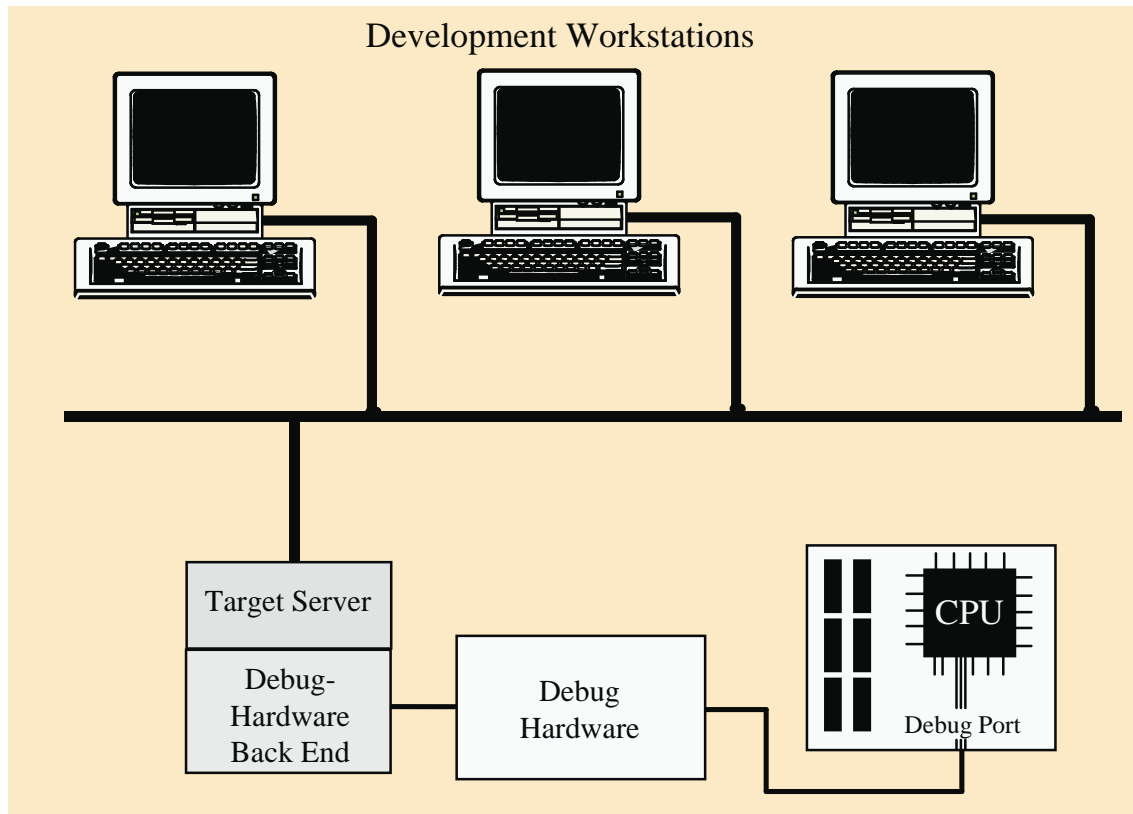


Figure 3. Tornado's client-server architecture lets a number of developers access the same target across a network

time and user-interface functions, this approach raises the cost.

## REAL-TIME CONTROL

An RTOS provides the control that an embedded system needs, ensuring that a task which draws graphics on the screen does not take precedence over one controlling the GSM protocol. To reduce development time and make it easier to provide more advanced applications, Wind River has worked with third parties to ensure that necessary Web browser software and communications protocols are available for Tornado.

As many embedded systems developers create and test the user-visible applications for their products using a desktop operating system such as Windows, Wind River has made it easier to transfer those applications to the target by partnering with Willows Software. Using the Willows RT for Tornado toolkit, developers can take the source code for applications that they have written to the Win32 interface and transfer them to a target that runs under VxWorks. Products developed using the toolkit can run on a wide range of microprocessors, including 68K, CPU32, PowerPC, i960, x86, SPARC and MIPS.

The embeddable library is based on a set of source code components that can be compiled and linked for the target environment, providing maximum portability. Applications compiled for the library run at native speed and can, like standard Windows applications, call API functions, receive messages, load resources and launch other Windows applications and DLLs.

All of these extra functions are not part of the core VxWorks operating system but take advantage of its dynamic structure. That gives developers the opportunity to scale software to suit individual designs and does not burden the system with unnecessary functions.

## SUMMARY : THE IMPORTANCE OF THE NETWORK

Networks are shaping embedded systems at all levels, from development through deployment to product design. Tornado's dynamic structure and its direct support for networking means that embedded systems developers can use network technologies where they are most appropriate without burdening the system if the final design does not need to be network attached. Where network technologies are helping to shape embedded systems, Tornado builds in support for the additional functions that OEMs need to get their products to market faster. Tornado is the link between real-time and the network.

*Claude Garcia, after his Master in Bio-Medical in 1981, worked for Thomson SDC as a Software Development Engineer on Counter measurement applications. He became a teacher for AFPA (Association pour la Formation Professionnelle des Adultes) and finally, joined Wind River Systems in 1990, where I've been a Field Application Engineer. Today Claude is the Technical Marketing Engineer for Europe, with 14 years experience in the Real Time world.*