Connect via Embedded Linux and Java

Abstract

This is a survey about the networking related features provided by embedded Linux. In this survey, we will summarize the major reasons why Linux is increasingly pervasive in embedded systems market. Seeing the trend that "everything to be connected" , we will focus on the networking related features provided in embedded Linux, and compare several distributions. We will also put an eye on the use of Java in embedded systems

1. *Introduction*

Linux is winning in the embedded OS. There are several major reasons:

- No per-unit royalty fees or licensing charges
- Providing more features existing RTOSes don't, such as connectivity, memory protection, security, and support for customizable color GUIs.
- Being Extremely flexible that you can definitely find some form of Linux to run on any processor
- Provide a single platform from PDA to clustered super computer to minimize the system specific issues

Since the embedded systems are increasingly shipped with more memory, more intelligent features, and are likely to be interconnected somehow in the future, they definitely need a more general purpose embedded OS instead of the traditional single-purpose device-specific OSes. As the reasons listed above, Linux has become prevailing in this area. The trend of embedded devices is to be more intelligent. In order to achieve the intelligence, embedded systems needs more connectivity, both among the devices and to the remote server. Therefore, we will focus on the connectivity an embedded Linux can provide.

2. *Comparison among Linux*

To be a good embedded OS for networking purpose, the real-time performance is also quite important, because networking always comes with asynchronous response-time-limited events. Here is a comprehensive real-time features comparison table from[1]. From this table, we can see some common mechanisms to achieve better real-time performance, such as " Fully preemptable" kernel[2], Precise scheduling.

From the functionality point of view, embedded Linux provides the same capabilities as normal Linux, and is very configurable. Since Linux has implementation of almost any kind of network function or protocol, it's easy to build the same functionality into any embedded version as long as the footprint is acceptable.

	Commercial Real-time Distributions				Open Source Real-time Projects						
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Separate real-time kernel	with RTLinux	with RTAI	with RTAI	with RTAI	✓	✓	✓				
Dual space programming	✓	✓	✓	✓	(2)		(1)	✓	✓	✓	✓
Improved Linux kernel preemption	✓	[illegible]		✓				✓			✓
"Fully preemptable" Linux kernel	announced	announced									
Precise scheduling within Linux	✓	[illegible]		✓				✓	✓	✓	
Includes solution for priority inversion within Linux		[illegible]									
"Hard" interrupt response, [illegible]	with RTLinux	with RTAI	with RTAI	with RTAI	✓	✓	✓				
"Moderate" interrupt response, [illegible] within Linux	✓	[illegible]		✓				✓			✓
Improved precision for timeliness, within Linux		[illegible]		✓				✓	✓		
NOTES: (1) = Limited (2) = Very limited											

3. *Java + Linux*

When talking about networking and program language, we can never ignore Java, which is born to the mission. Java is perfect for handling pure networking issues, and for the exponentially growing number of embedded devices, Java seems to be the simplest and quickest way to develop and deploy applications. As the release of J2ME (Java2 Micro Edison), lots of drawbacks of Java in case of using it in embedded system are overcome, e.g. the footprint size can be as little as 128K for VM and libraries[3]. The real-time specification for Java enables processes to interrupt the garbage collector to ensure predictable response amongst many other improvements. AOT (Ahead-of-Time), JIT (Just-in-time) and Dynamic Compilation speed up Java dramatically. Here is a comparison among these techniques [4]:

Feature	Speed	ROM	RAM
Interpreter	Low	Low	Low *
JIT Compiler	Med	Med	High
Adaptive JIT	Med+	Med+	Med
AOT Compiler	High	High	Low

* Med, if byte codes don't execute from ROM

Everything is showing that Java is ready to go for embedded systems, and the combination of Java and Linux will maximize the flexibility and minimize the development and maintenance.

4. *Future work*

1) I'll compare the network performance of two embedded Linux with C/C++ language.

2) I'll compare the implementation detail of both embedded Linux.

3) I'll build a test-bed on a selected Linux, and compare the performance of different Java compilation techniques on it and on Windows.

4) Finally, I'll try to build a simulated intelligent devices network to demonstrate the inspiring future of Linux and Java.

---

[1] *Kevin Dankwardt, Comparing real-time Linux alternatives,http://www.linuxdevices.com/articles/AT4503827066.html*

[2] *George Anzinger and Nigel Gamble, Design of a Fully Preemptable Linux kernel, http://www.linuxdevices.com/articles/AT4185744181.html*

[3] *J2ME(TM) Connected Limited Device Configuration (CLDC) ("Specification")*

[4] *Randy Rorden, Java and Embedded Linux Team Up, http://www.linuxdevices.com/articles/AT7873839273.html*