CILIX: a Small CIL Virtual Machine for Wireless Sensor Devices

Yasue Kishino[1], Yutaka Yanagisawa[2], Tsutomu Terada[3], Masahiko Tsukamoto[3], and Takayuki Suyama[1]

[1] NTT Communication Science Laboratories, NTT Corporation, Japan
[2] Nippon Telegraph and Telephone West Corporation, Japan
[3] Graduate School of Engineering, Kobe University, Japan

Abstract. In this paper, we propose a small virtual machine called CILIX, which can execute Common Intermediate Language (CIL) executable codes. Although we have designed CILIX to require very small resources, it supports threading, wireless data transferring, and two ways of rewriting executable codes: over-the-air programming and use of a microSD card. We can develop executable codes for small devices using our favorite programming language.

1 Introduction

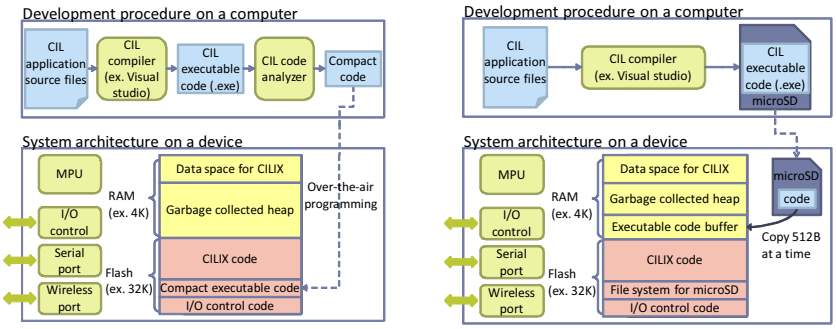
Many small wireless computer devices have been developed for constructing sensor networks. Small devices have few sensors and can be operated with small batteries. Mica MOTE [2] and smart-It [1] are well known examples of such small devices. Most of these devices are designed individually for specific purposes, and we consider that this individuality makes it impossible to standardize their specifications. In the future, we may have to construct a sensor network with various types of small devices.

The cost of software development for heterogeneous networks continues to increase. If a network consists of a hundred different devices, we have to learn how to program each of them. Before developing software, we must also create a hundred programming environments in which to develop the devices.

Virtual machines constitute an effective way to avoid the increased cost needed to develop software for various hardware architectures. We can develop, test, and debug software for every device in the same development environment for a virtual machine.

There is a virtual machine called SimpleRTJ [5] which enables us to use Java to write program codes for such small devices. The runtime system of SimpleRTJ requires only 20 KB RAM and a 16bit MPU. SimpleRTJ appears to avoid the problem of having to develop programs for heterogeneous sensor networks, but the following two problems remain with this virtual machine: (1) If the developer does not know Java, he/she must learn it before developing a program code. (2) It is hard to replace a program code once the code is embedded in a small device.

When we rewrite an executable code on a device, we must directly replace the code in the ROM of the device by connecting it to a PC. Although we often replace the executable code on devices to improve the performance of a sensor network, the executable code is not rewritable on the existing virtual machine.



(a) Over-the-air programming (b) MicroSD programming

Fig. 1. Development procedure and system architecture

To overcome these problems, we have developed CILIX. It is a small powerful virtual machine which can understand CIL [3] executable codes. CIL is an open specification (published under ECMA-335 and ISO/IEC23271), in which applications written in multiple high-level languages can be executed in different system environment. We solve the first problem by providing a "multi-lingual" virtual machine. On the other hand, we solve the second problem by using over-the-air programming facility and microSD support.

2 CILIX

CILIX requires at least a 32KB program memory, 4KB of working RAM, and a 16bit MPU. We consider this requirement to be reasonable for such existing small devices as Mica MOTE and Smart-It. Figure 1 shows the development procedure and system architecture of CILIX. We have mounted CILIX on MSP430, NEC V850, and Windows PC. Figure 2 and 3 show one of the prototype devices we used for this demonstration. The prototype device consists of an MSP430 micro controller, a temperature sensor, a light sensor, a 315MHz wireless communication module, and a serial communication port, which can be connected to an EL panel. We describe our small virtual machine CILIX in detail.

Multi language development Since CILIX is compliant with CIL, developers can write programs using various languages that have a CIL compiler. We have evaluated several types of software which are written by Managed C++, Visual Basic, and C# (Figure 4). The developers can write, test, and debug program codes on existing powerful IDEs such as Visual Studio .NET. The executable code can operate on both CILIX and a PC that supports CIL. Moreover, the developer can choose the Mono [4] and Linux OS for the development.

Saving device resources Small devices have limited memories. Although we optimized CILIX for small programs, it supports most CIL instructions which are required for sensor data processing.

Portability We developed CILIX by employing C, which we can use with most micro controllers. The core of CILIX and I/O control code, which depends on

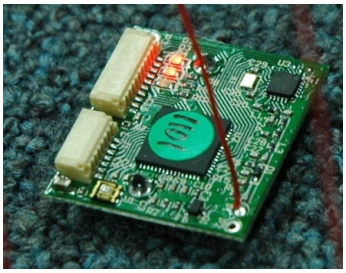


Fig. 2. Prototype device

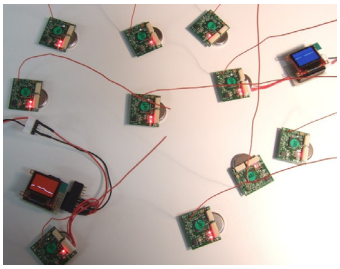


Fig. 3. Demonstration

```
1.  using System;
2.  using System.Threading;
3.
4.  namespace Sroom
5.  {
6.    class ThreadedRF
7.    {
8.      Thread thread;
9.      int sensorValue;
10.     public bool quit = false;
11.     public void Start()
12.     {
13.       thread = new Thread(new ThreadStart(ThreadMethod));
14.       thread.Start();
15.     }
16.     public void ThreadMethod()
17.     {
18.       while(!quit)
19.         lock(thread)
20.         {
21.           short version = 0;
22.           byte[] data = Cilix.ReceiveWireless(ref version);
23.           if(version == -1)
24.           {
25.             // process received data
26.           }
27.           else if(version>3 && sensorValue>150)
28.             Cilix.UpdateProgram(version);
29.           else
30.             Cilix.IgnoreProgram(version);
31.         }
32.     }
33.     public void Sensing(int n)
34.     {
35.       for(int i = 0; i < n; i++)
36.       {
37.         lock(thread)
38.         {
39.           sensorValue = Cilix.GetTemperature();
40.         }
41.         // process sensor data
42.       }
43.     }
44.     [STAThread]
45.     static void Main()
46.     {
47.       ThreadedRF th = new ThreadedRF();
48.       th.Start();
49.       th.Sensing(100);
50.       th.quit = true;
51.     }
52.   }
53. }
```

Fig. 4. Sample code of wireless programming and sensing (C#)

the micro controller hardware, are separated. We can easily port CILIX to the other micro controllers. Actually the I/O control code was accounted for 0.8% of CILIX in MSP430.

Executable code replacement via wireless communication Developers can repurpose CILIX devices by replacing the executable code via wireless communication. The over-the-air programming facility makes it easy to replace the executable codes at many sensor nodes and at sensor nodes allocated a location we cannot pick up directly. Figure 1(a) shows this architecture. Since Flash memory size is limited, we send compact executable codes that do not contain unnecessary parts of the original executable codes.

Moreover, CILIX provides rewrite control functions. In Figure 4, the executable code can determine to accept the new code by itself (lines 27 to 30). For example, we can overwrite a new program for detailed processing at sensor nodes allocated to a high temperature area.

When a sensor node receives a new executable code and it determines that the code should be overwritten, it stops the current program. The new code is transmitted by several packets, and the new code is overwritten to the program code area where the current file is stored. Once every part of the new code has been stored in the program code area, CILIX starts executing the new code.

Executable code replacement using microSD card CILIX also supports microSD programming for sensor nodes without a wireless communication function or debugging. Figure 1(b) shows this architecture. CILIX can execute an executable code placed on a microSD card.

Other functions CILIX has a garbage collected heap and supports the threading (In Figure 4 two threads are running.) and transferring of data using wireless communication.

3 Demonstration

In our demonstration we show the dynamic replacement of executable codes depending on the sensor value. Figure 3 shows an example demonstration. At the beginning of the demonstration, all sensor nodes behave similarly. After we have rewritten the executable codes on some sensor nodes, which are allocated bright areas, their behaviors change dynamically and they restart sensing more frequently.

We show also an actual development using several programming languages.

4 Conclusion

We introduced our small and powerful virtual machine named CILIX. CILIX can understand CIL executable codes and enables us to develop by various programming languages. Our small virtual machine supports two way of rewriting executable codes: over-the-air programming and use of microSD card. We plan to construct a data aggregation sensor network using hundreds of CILIX devices, and a kind of life-log application that records human daily activities.

Acknowledgement

Part of this work was supported by JSPS Grant-in-Aid for Scientific Research (A) (20240009).

References

1. M. Beigl and H. Gellersen: Smart-Its: An Embedded Platform for Smart Objects, Smart Objects Conference (sOc) 2003, 2003.
2. S. Madden, R. Szewczyk, M.J. Franklin, and D. Culler: Supporting Aggregate Queries Over Ad-hoc Wireless Sensor Networks, in Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 49–58, 2002.
3. Standard ECMA-335 Common Language Infrastructure (CLI), http://www.ecma-international.org/publications/standards/Ecma-335.htm .
4. Mono project, http://www.mono-project.com/Main_Page .
5. The Simple Real Time JAVA, http://www.rtjcom.com/ .