Diss. ETH No. 21182

Compiler-Assisted Thread Abstractions for Resource-Constrained Systems

DISSERTATION
submitted to
ETH ZURICH
for the degree of
DOCTOR OF SCIENCES
(Dr. sc. ETH Zürich)

by

ALEXANDER BERNAUER
Diplom-Informatiker, University of Ulm, Germany
born September 11, 1980
citizen of Germany

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Kay Römer, co-examiner
Prof. Dr. Thomas Gross, co-examiner

Abstract

Although increase of hardware resources is the general trend in information technology, resource-constrained systems continue to be of relevance. In wireless sensor networks (WSN), for example, a typical device is equipped with a few kilobytes of RAM and provides only limited computing power. Such systems aim to provide ad-hoc and long-term monitoring of physical or environmental conditions over large spatial regions. Maximizing the number of participating devices and extending the overall network lifetime is crucial for such applications. Moore's Law is therefore exploited towards lower device costs and lower power consumption at almost constant resource capabilities per device.

This scarcity of resources has various implications on how resource-constrained systems such as wireless sensor networks are designed and implemented. Besides requiring highly optimized network protocols, one particularly important consequence is the prevalent adoption of the event-based programming paradigm. This paradigm is considered well-suited for traditional WSN applications, because it can be implemented efficiently and matches the reactive nature of such applications. However, requirements of WSN applications are continuously increasing, causing higher software complexity to meet growing expectations. As a result, today's sensor networks often run IP stacks, HTTP and CoAP services, middleware, and other extensive software components. Implementing the complex control and data flows of such applications and services in an event-based manner is error-prone and the resulting code is hard to maintain. This leads to software faults that are costly and time consuming to track down, both during development and testing, and particularly during deployment and operation. Additionally, the presence of software faults entails security issues for a deployed system. To counter these problems, better programming abstractions for resource-constrained systems seem necessary.

Threads are known for overcoming many problems of events by supporting sequential control flows via synchronous functions, but existing solutions either provide incomplete thread semantics or introduce a significant resource overhead. This reflects the common believe that expressiveness has to be traded for efficiency and vice versa. With our work, we show that this trade-off is not inherent to resource-constrained systems.

We follow the approach of compiler-assisted thread abstractions, where full-fledged thread-based C code is compiled to equivalent event-based C code that runs atop an event-based operating system. This approach is promising for two reasons. First, the event-based run-time system avoids multiple preallocated stacks and the overhead of a task scheduler. And second, a compiler can, in contrast to a run-time-based thread library, perform static program analysis and thus apply application-specific optimizations.

Our thesis is that *a comprehensive thread abstraction is possible also for resource-constrained systems. A compiler, which translates thread-based applications into event-based programs, is able to combine the efficiency of event-based programming with the comfort of thread-based programming. This addresses the increasing requirements of resource-constrained systems.* We develop our thesis in the context of wireless sensor networks, assuming that resource scarcity will continue to be relevant. Existing threading

solutions either provide only limited thread semantics or require more resources than common WSN devices offer. We therefore propose a compiler-assisted approach that can provide a comprehensive and efficient thread abstraction.

We support our thesis by presenting a comprehensive compiler-assisted thread abstraction concept for resource-constrained systems. To achieve this, we developed a platform-agnostic code transformation from thread-based C code to equivalent event-based C code. This enables developers to write thread-based applications without the additional costs of a thread library. Moreover, we present a way to reverse this transformation at run-time for fault diagnostics purposes. Hiding the details of the transformation and the underlying run-time system during all phases of development completes the abstraction. In order to evaluate our approach, we designed and implemented a compiler and a debugger prototype to conduct a set of experiments. Our results show that compiler-assisted thread abstractions not only outperform thread libraries, but are even almost as efficient as hand-written event-based code. The identified overhead accounts for only 1% higher RAM usage, 2% additional processor cycles, and 3% larger binaries on average, which we consider a reasonable trade-off for the gained comfort of a comprehensive thread abstraction. Additionally, the experiments demonstrate that sustaining the abstraction level for fault diagnostics is possible.

Zusammenfassung

Obwohl der allgemeine Trend in der Informationstechnik in Richtung immer leistungsfähigerer Hardware geht, bleiben ressourcenbeschränkte Systeme weiterhin von Relevanz. Drahtlose Sensornetze, zum Beispiel, verwenden typischerweise Rechnerknoten, die mit wenigen Megahertz getaktet werden und nur über einige Kilobyte Speicher verfügen. Der Grund dafür ist, dass solche Systeme eine infrastrukturlose, langfristige und geographisch ausgedehnte Überwachung von physikalischen Phänomenen ermöglichen sollen. Da hierfür eine Maximierung der Anzahl der Knoten und der Lebensdauer des Systems entscheidend ist, wird der durch das Mooresche Gesetz induzierte Effizienzgewinn für niedrigere Gerätekosten und niedrigeren Energieverbrauch bei annähernd gleichbleibenden Hardwareressourcen pro Gerät genutzt.

Die Ressourcenknappheit hat einen starken Einfluss auf Entwurf und Implementierung solcher Systeme, so dass es zum Beispiel für Sensornetze hoch optimierter Kommunikationsprotokolle bedarf. Ausserdem ist in dieser Domäne das ereignisbasierte Programmiermodell weit verbreitet, da sich dieses Paradigma effizient umsetzen lässt und es gut zur reaktiven Natur traditioneller Sensornetzanwendungen passt. Die Anforderungen an solche Anwendungen nehmen allerdings kontinuierlich zu, was zu einer erhöhten Komplexität der Programme führt. Um den steigenden Erwartungen Rechnung zu tragen, werden auf heutigen Sensorknoten u.a. IP-Protokollstapel, HTTP- und CoAP-Dienste, Middleware und weitere umfangreiche Softwarekomponenten betrieben. Die ereignisbasierte Umsetzung von Kontroll- und Datenflüssen solcher Anwendungen ist allerdings fehleranfällig, und die Pflege entsprechender Software gestaltet sich als schwierig. In der Konsequenz führt dies zu Softwarefehlern, deren Analyse und Behebung während Entwicklung und Erprobung, aber vor allem während Inbetriebnahme und Betrieb kostspielig sind. Darüber hinaus können solche Softwarefehler auch sicherheitsrelevante Schwachstellen sein. Um all dem entgegenzuwirken, sind höhere Programmierabstraktionen für ressourcenbeschränkte Systeme nötig.

Threads können durch sequenziellen Kontrollfluss und synchrone Funktionen viele Probleme der ereignisbasierten Programmierung beheben. Bei bereits existierenden Lösungen ist allerdings entweder die unterstützte Threadsemantik eingeschränkt oder es werden leistungsfähigere Systemkomponenten benötigt als solche, die typischerweise in drahtlosen Sensornetzen zum Einsatz kommen. Das überrascht nicht, da es der allgemeinen Meinung entspricht, dass es einen Zielkonflikt zwischen der Vollständigkeit einer Threadsemantik einerseits und der Effizienz ihrer Implementierung andererseits gibt. Unsere Arbeit zeigt allerdings, dass dieser Zielkonflikt keine inhärente Eigenschaft von ressourcenbeschränkten Systemen ist.

Dabei verfolgen wir den Ansatz der übersetzergestützten Threadabstraktion. Ein Compiler wandelt hierzu ein vollwertiges threadbasiertes C-Programm in ein äquivalentes ereignisbasiertes C-Programm um, welches von einem ereignisbasierten Betriebssystem ausgeführt wird. Dieser Ansatz ist aus zwei Gründen vielversprechend: Erstens ist das übersetzte Programm ereignisbasiert, so dass sowohl vorbelegte Stacks als auch der zusätzliche Aufwand eines Thread-Scheduler vermieden werden. Und zweitens kann ein

Compiler, im Gegensatz zu einer laufzeitbasierten Threadbibliothek, statische Programmanalysen durchführen und somit anwendungsspezifische Optimierungen vornehmen.

Unsere These ist daher, *dass eine umfassende Threadabstraktion auch auf ressourcenbeschränkten Systemen möglich ist. Ein Compiler, der threadbasierte Programme in ereignisbasierte Programme umwandelt, ist in der Lage, die Effizienz von ereignisbasierter Programmierung und den Komfort threadbasierter Programmierung zu vereinen. Insgesamt kann so den wachsenden Anforderungen an ressourcenbeschränkten Systemen Rechnung getragen werden.* Wir entwickeln unsere These im Kontext von drahtlosen Sensornetzen, wobei wir davon ausgehen, dass Ressourcenknappheit auch in Zukunft relevant sein wird. Existierende Threadlösungen aus dieser Domäne benötigen entweder mehr Ressourcen als ein typischer Rechnerknoten zur Verfügung stellt oder sie unterstützen nur eine eingeschränkte Threadsemantik. Wir schlagen daher vor, mit Hilfe des übersetzergestützten Ansatzes eine umfangreiche und dennoch effiziente Threadabstraktion zu verwirklichen.

Wir untermauern unsere These dadurch, dass wir ein Konzept für eine übersetzergestützte Threadabstraktion für ressourcenbeschränkte Systeme angeben. Zu diesem Zweck haben wir eine plattformunabhängige Transformation von threadbasierten C-Programmen zu äquivalenten ereignisbasierten C-Programmen entwickelt. Entwickler können damit threadbasierte Anwendungen ohne die zusätzlichen Kosten einer Threadbibliothek realisieren. Ausserdem zeigen wir auf, wie man zum Zwecke der Fehlersuche die Transformation zur Laufzeit rückgängig machen kann. Indem wir damit die Details der Transformation und der darunterliegenden Laufzeitumgebung während allen Entwicklungsphasen verbergen, vervollständigen wir die Abstraktion. Zur Bewertung dieses Ansatzes haben wir einen Prototypen eines Übersetzers und eines Debuggers entwickelt und damit eine Reihe von Experimenten durchgeführt. Unsere Messungen haben ergeben, dass übersetzergestützte Threadabstraktionen hinsichtlich der Effizienz nicht nur Threadbibliotheken übertreffen, sondern auch fast so effizient wie handgeschriebene ereignisbasierte Programme sind. Der gemessene durchschnittliche Mehrverbrauch an Ressourcen beträgt 1% RAM sowie 2% Prozessorzyklen und resultiert in 3% grösseren Binärprogrammen. Unserer Meinung nach sind diese geringen Mehrkosten aufgrund des gewonnenen qualitativen Mehrwerts einer umfangreichen Threadabstraktion gerechtfertigt. Ausserdem zeigen die Experimente, dass es möglich ist, die Abstraktion auch während der Fehlersuche zur Laufzeit aufrechtzuerhalten.