FAKULTÄT FÜR !NFORMATIK

Master-/Diplomstudium:
Technische Informatik

Diplomarbeitspräsentationen der Fakultät für Informatik

Design and Implementation of TinySpaces
The .NET Micro Framework based Implementation of XVSM for Embedded Systems

Alexander, Marek

Technische Universität Wien
Institut für Computersprachen
Arbeitsbereich: Programmiersprachen und Übersetzerbau
BetreuerIn: A.o. Univ. Prof. Dr. Dipl.-Ing. eva Kühn

Context

There is a continuously growing amount of embedded devices which need to collaborate in some networked environment thereby using different communication media like in the area of building automation and industrial automation. Such systems are also called **networked embedded systems.** (NES)

Today's communication middlewares are generally based on **CORBA** and do not allow for **P2P communication**. However this paradigm is very useful in scenarios, where the devices are mobile. (mobile agents)
Some examples are:

- **Intelligent Transport Systems** (ITS) where moving vehicles communicate with their static environment.
- **Robotic vacuum cleaners** which collaborate in a P2P fashion to partition the work among another or to navigate through an unknown environment.

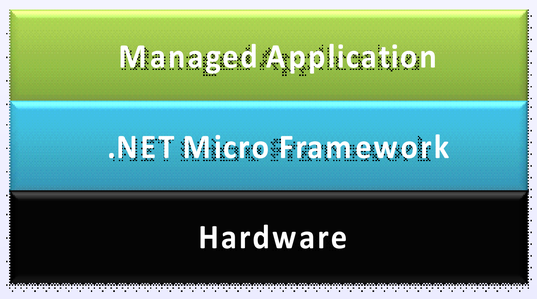
Therefore, there is a growing need for a P2P middleware to interconnect those devices. Using a P2P approach can furthermore result in a better scaling of a networked embedded system than a centralized one.

For the implementation of **TinySpaces,** the first prototype of a XVSM middleware for resource constrained devices, the .NET Micro Framework was chosen.

The **.NET Micro Framework** is a bootable run-time and a development environment for resource-constrained devices.

It brings many benefits to the world of embedded development:
- Managed code (garbage collector)
- Modern programming language (C#)
- An adapted subset of the .NET base class library
- Live-debugging support
- Etc.

The main benefit however is that code written for .NET MF is **platform independent** and already supported by numerous devices.



Furthermore the great tooling support allows for rapidly developing prototypes of P2P based networked embedded systems to evaluate this technology.

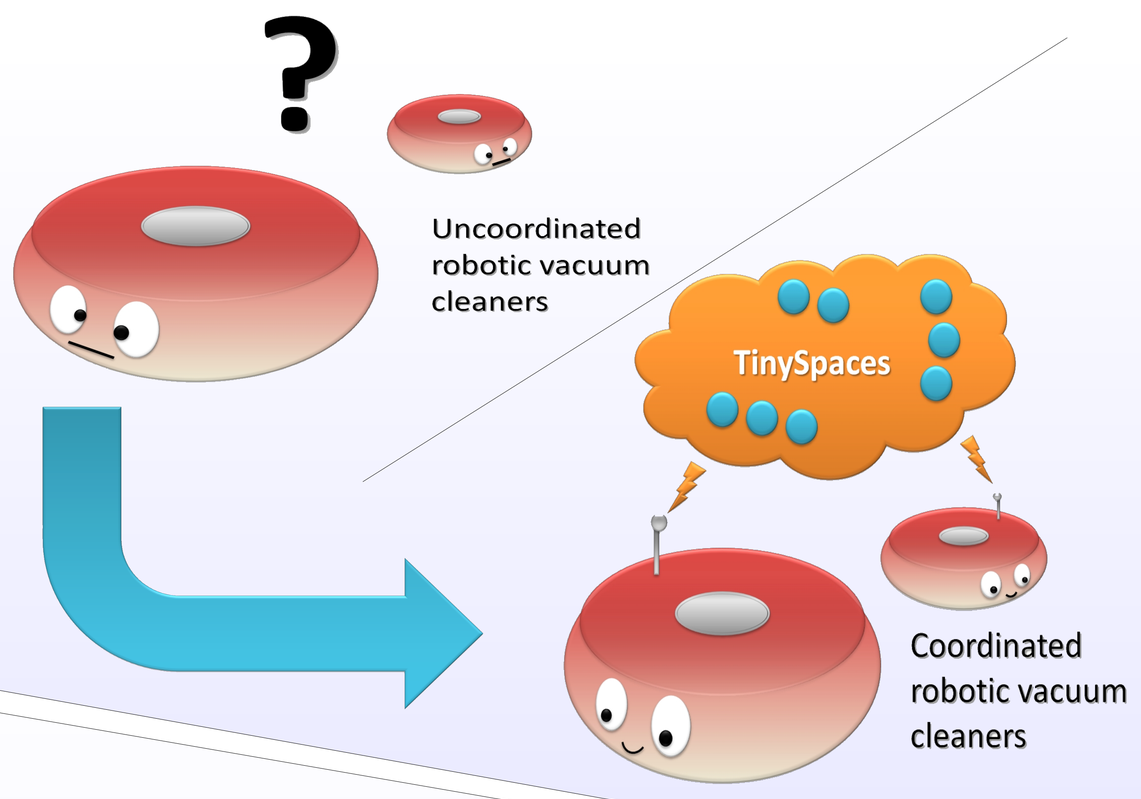
Challenge

TinySpaces need to meet the requirements of embedded devices while complying with the definition of XVSM (or a subset) to remain compatible to other implementations of XVSM

Requirements of NES
- Minimum code-size
- High run-time efficiency
- Low power consumption
- Minimal packet sizes of messages.

VS.

XVSM specification
- Layered architecture
- Transactions
- Multiple coordinators and selectors
- Aspects
- Concurrent requests
- ...



Results

TinySpaces where developed using Contract First Design following the layered architecture of XVSM → Layers can be replaced.

Layer 5 – Run-time
- Supports concurrent requests, timeout handling (sollback of expired transactions) and wait-handling (scheduling)
- No polling and busy waiting → higher energy efficiency

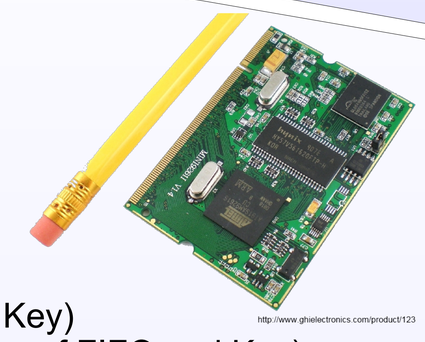
Layer 5b - Communication
- Multiple transport services at the same time.
- Supports multiple serialization mechanisms which can be configured per service instance. Three mechanisms provided:
  - ***Binary****:* Best performance; lowest byte usage; smallest code-size; platform dependent.
  - ***Binary Interop****:* Average performance; highest byte usage; supported by all .NET platforms; average code-size
  - ***XML****:* Platform independent; lowest performance; average byte usage; largest code-size

Layer 4 - Aspects
- Space aspect support -> Notification aspect provided
- Container aspect contracts available → not implemented for smaller code-size and higher run-time efficiency. Could be implemented easily.

Layer 3 - Coordination
FIFO- and Key Coordinator/Selector provided. Support for combination of multiple selectors and queries for a maximum of expressiveness.

Layer 2 – Transactions
- Implicit/explicit transactions (each operation with transaction)
- Pessimistic locking and isolation level Read Committed
- Entry and Container level locking

Layer 1 - Basic Operations
- Query support
- Atomic operations: **read**, **write**, **take**, **destroy**



The benchmarks were performed using ChipworkX Development board V1.2 of GHI electronics (200 Mhz 32-bit ARM 9 Processor along with 64MB SDRAM) running .NET Micro Framework 3.0:

- Run-time efficiency
  - ~60 writes per second (avg. of FIFO and Key)
  - ~23 reads/takes/destroys per second (avg. of FIFO and Key)
  - ~6 MB (100 containers with 100 entries each) - logarithmic scaling
- Power consumption not testable as device does not support power saving features yet. (CPU suspension, etc.)
- Average packet size of 73,3 bytes (Binary)
- Minimal code-size of TinySpaces: 205 KB

Conclusion

Developing TinySpaces, several compromises had to be made to satisfy the requirements of networked embedded devices as well as to comply with a subset of the XVSM specification.
Nevertheless it could be shown, that space-based computing can also be achieved with resource-constrained devices.
Following the layered approach in combination with contract first design allows for customizing the space to adapt it for specific application areas.

Kontakt: e0726166@student.tuwien.ac.at