Enabling Smart Buildings

Security, Autonomy and Services

Dr. Christoph Niedermaier, Siemens CT
Holger Kinkelin, TUM
Andreas Müller, TUM
PART I

Autonomy and Services
Motivation for Model-based Approach

- **Motivation**: Supporting heterogeneous devices and services
  - Specification of messages and message groups (topics)
  - Development of new services using existing messages
  - Deployment of existing services to new devices

- **Approach**:
  - Model-based generation of (WSN) applications
  - Model-based translation of message representations
  - Model-based application-independent components
  - Platform independence (Web-oriented, standard tools)

- **Benefits**:
  - Better time to market (rapid application development)
  - Minimization of bugs at development time
  - Reuse of mechanisms and components ("services")
Problem: Heterogeneous Devices and Services

- Service deployment on (WSN) devices?
  - Monolithic applications → no dynamic deployment
  - Static component model (TinyOS) requires wiring

- Optimization of development effort?
  - Commonalities:
    - Messaging (event channels ≈ topics, event scope ≈ range / priority / domain)
    - Persistence (event log)
    - Publish/Subscribe (topics)
    - Management (services / devices)
  - Variabilities:
    - HW platform (sensors, actuators, flash, …)
    - Applications (services, messages, subscriptions, publications, …)

```
<service name="BlinkReporter">
  <description>
    Service that reports blink
  </description>
  <subscriptions>
    <topic name="BlinkInfo"/>
  </subscriptions>
  <publications>
    <topic name="BlinkControl"/>
  </publications>
</service>
```
Basis: Autonomic Service Framework

Control of devices / services
- Device shutdown / reboot
- Service activation
- Service discovery

Control of message streams
- Dispatching
- Logging
- Persistence
- Priority
- Scope

Logging of messages
- Transient in RAM
- Persistent in flash

Different realizations
- Scoped “broadcast”
- Tree routing towards gateway
Model-based Code Generation

```
<service name="BlinkReporter">
  <description>
    Service that reports blink events.
  </description>
  <subscriptions>
    <topic name="BlinkInfo" />
  </subscriptions>
  <topic name="BlinkControl">
    <description>
      Channel for notification of blink events.
    </description>
    <keys>
      <key name="BlinkSignal" />
    </keys>
    <scope name="LocalEvent" />
    <buffer size="1" blockcount="0" />
  </topic>
</service>

class ASFGenerator:
  def __init__(self, model, factory, host='localhost',
               self.model = model
               self.factory = factory
               self.host = host
               self.user = user
               self.password = password
               self.database = database

module BlinkReporterF:
  provides interface Init as ServiceInit;
  provides interface SplitControl as ServiceControl;

  uses interface ServiceInfo;
  uses interface Query;
  uses interface Store;

  uses interface Subscribe as SubscribeBlinkInfo;
  uses interface Publish as PublishBlinkControl;

  TOSMAKE_PATH := $(XDIR)/support/make
  XI_PLATFORMS = zl win32
  ifneq ($(findstring sim, $(MAKECMDGOALS)),sim)
    CPLASS += -Os
  endif
```
ASF Model – The Model Building Blocks

- Constants
- Types
- Events
- Services
- Hardware
- Platforms
- Nodes
- Deployments
From Model to Application: Sample Service Model

XML

```xml
<key name="BlinkTrigger" id="11">
  <description>
  Message for triggering blinking behaviour
  </description>
  <record>
    <field name="period" type="blink_period_t"/>
    <field name="action" type="blink_action_t"/>
    <field name="color" type="blink_color_t"/>
  </record>
  <scope name="BlinkEvent"/>
</key>
```

Code from app developer

```python
def BlinkReporterC()
    "Provide interface Init as ServiceInit;
    provides interface SplitControl as ServiceControl;
    uses interface ServiceInfo;
    uses interface Subscribe as SubscribeBlinkInfo;
    uses interface Publish as PublishBlinkControl;
    uses interface Query;
    uses interface Store;

nesC (TinyOS): Wiring (generated)

```python
"struct key_blink_trigger_{
    blink_period_t period;
    blink_action_t action;
    blink_color_t color;
} __attribute__((packed));

nesC: Header file (generated)
Model-based Code Generation – The Benefits

- Better time to market
- Reduced coding error rate
- Reuse of mechanisms / components → TinyOS component model
- Reduced effort for testing and documentation
- Platform independence
  - Visualization clients (browser based)
  - Messaging clients (XMPP based)
  - (Free) Standard software (Apache, MySQL, PHP, JavaScript)
- Abstraction facilitated by generic data structure
  - Use of generic clients (Web tools) is supported
  - Client applications can be (partly) generated automatically
PART II

Security
Motivation: Trends in Home Networking

- Growing number of entities in homes:
  - networked devices
  - offered services
  - users

- Convergence of multimedia, home automation, „classic“ networking

- Desire to share services between homes

- Security becomes important

Future security needs can not be satisfied by current home solutions
Building Blocks of our Security Architecture

- **Service for Local Identity Management**
  - User’s device is "paired" with the home and receives a unique identity
    - Identities are needed for accessing services
    - Identities are managed by the home
    - Each home can be seen as one trust domain

- **Service for Home Pairing**
  - Establishes trust relationships between homes / trust domains
  - Enables the sharing of e.g. services

- **Service for Rights Management**
  - Controls access to services

- **Secure hosting of services**
  - Our security services run in a shielded environment
Local Identity Management - Basics

- Certificate Authorities are often used in Enterprise networks to create secure identities for entities
  - CA „vouches“ for identity
  - Not directly applicable for home networks

- Certificates are used for authentication within the network

- Idea: Build assistance system that enables home users to run their own CA at home and issue certificates to local devices („pairing“)
Identity Management – Pairing mechanism

- We created an easy to use mechanism that “pairs” new devices with the home network.
  - User/owner of the device is identified implicitly
  - Side effect of the pairing mechanism is that the new device gets a valid certificate signed by the local CA
  - The new entity is added to a local Entity Directory that holds all known devices, services and access rights.

- Every step is controlled by the owner of the local network
Home Paring

- Home networks only know and trust their local CA
  - „Foreign“ Devices paired to a friend‘s network cannot be authenticated
    → No sharing of services

- Need to exchange certificates of the home networks among each other.

- We created two introducer mechanisms:
  - Direct – two members of the home network meet and exchange the public key of their Micro CA‘s
  - Indirect – the public keys are exchanged via the Internet
Rights Management

- Large scale networks often control access rights using the standardized and powerful XACML framework
  - Access rights are specified with XACML policies.

- We integrated XACML to our architecture and created easy to use interfaces to generate XACML policies.

- A policy generator translates the current settings in the Entity Directory to XACML policies

- XACML policies are consumed by authorization mechanisms
Secure Hosting of Services

- Virtualization isolates groups of services
  - Device pairing
  - Home CA
  - Management services
  - ...

- Two distinct virtual Wi-Fi networks offer access to the registration service and the service network

- Home CA leverages trusted computing technology to protect keying material
Conclusion

- Security needs will grow in home networks
- Current solutions can not be applied

Our solution:
- Integration of enterprise grade security solutions for identity management and authorization
- Special adoptions to unexperienced users that allow them to control security functionality
- Also useful in other network types, e.g. start-ups / small enterprises, etc
Enabling Smart Buildings

Thank you for your attention!

Questions?