



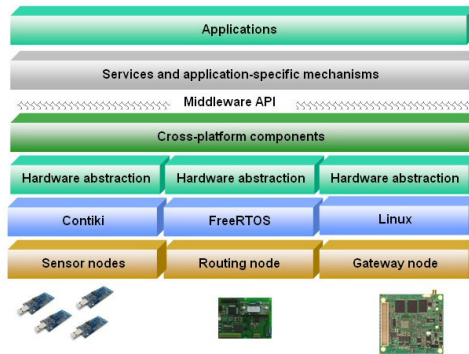
SOFTWARE PLATFORM

Reconfigurable Ubiquitous Networked Embedded Systems

Component-based Middleware

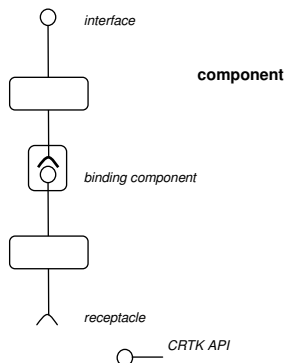
The RUNES middleware provides abstractions to simplify both application and system development. The middleware is built around Component Frameworks (CF). CFs are re-usable and dynamically-deployable software architectures. Highly modularised and customisable services and application-specific mechanisms can be built as compositions of components and/or CFs. These have been validated in the context of the 'fire in a road tunnel scenario' and deployed on devices ranging from PCs to PDAs to tiny sensor devices running the Contiki OS.

The RUNES middleware is based on a two-level architecture: The foundation is a minimal, language-independent, component-based programming model that is sufficiently lightweight to run on any of the devices typically found in networked embedded environments. Above this is a layer of software components that offer the necessary middleware functionality. Rather than provide a monolithic middleware 'layer', we separate orthogonal areas of middleware functionality into self-contained components that can be selectively and individually deployed according to current resource constraints and application needs. Crucially, the set of such components can be updated at runtime to provide the basis of a highly dynamic and reconfigurable system.



Component model design at a glance

- defines components as basic *run-time units*
- enables components to be instantiated *at run-time*
- functionality provided by components through *interfaces*
- dependancies expressed through *receptacles*
- receptacle/Interface binding made with *connector* components



Component

A component is an encapsulated unit of functionality and deployment e.g. components implementing routing protocols such as AODV or DSR, compression libraries, or even OS functions (power management or thread scheduling).

Interface

Interfaces make the functionality a component provides visible to other components.

Receptacles allow components exploit functionality provided by other components.

FP6 RUNES Integrated Project (IST 004536)

Overview

RUNES integrates and extends cross layered research ranging from hardware and network communication to application usability and middleware to advanced control theory and verification tools for validation of the methods and prototypes.

Objective

Provide an architecture, middleware and specialised simulation/verification tools that enables the creation of large-scale, widely-distributed, heterogeneous networked embedded systems that inter-operate and adapt to their environments.

Partners

- connectBlue
- CRC
- Ericsson
- IABG
- Kodak
- KTH
- Lancaster University
- LiPPERT
- Lund University
- NICTA
- Politecnico di Milano
- RWTH Aachen
- SICS
- Università di Pisa
- UCL
- UC Berkeley
- UC San Diego
- University of Patras
- University of Queensland
- Victoria University

Countries involved

- Australia
- Canada
- Germany
- Greece
- Italy
- Hungary
- Sweden
- UK
- USA

Start of the project

Sept 2004

End of the project

July 2007

Interaction

Allows components on different hosts to interact with loosely coupled interaction styles such as tuple spaces or the publish-subscribe model.

Discovery and advertising

Enables discovery of new services so that they can be found by other nodes.

Logical mobility

Injects new application functionality on devices as new requirements are realised.

Sensor co-ordination

Synchronises sampling periods and co-ordinate sleep periods when the sensing devices turn their radio off.

Context sensing

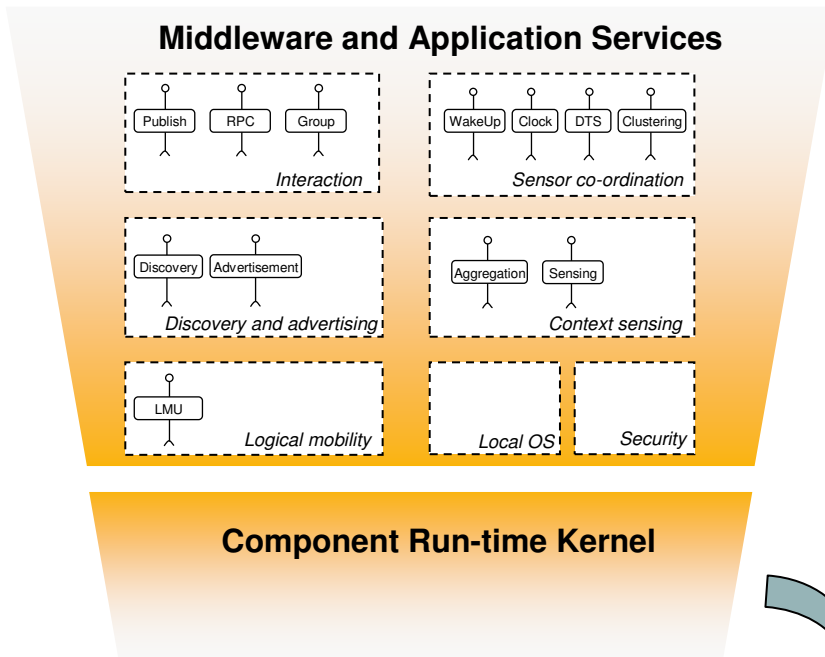
Senses the context in which the system is operating and provides this information to other components so that they can adapt their behaviors.

Security

Addresses a number of trust and privacy issues related to dynamically sending and receiving code.

Local OS

Realises an abstraction layer over operating system functionality, e.g. components implementing different scheduling policies or memory management techniques.



Component run-time Kernel

Manages the different entities defined in the component model at run-time in terms of loading and instantiating components and binding their interfaces. By providing different implementations of the kernel for a target hardware platform, heterogeneous, networked embedded systems are realised.

Kernel Implementations

The core abstractions defined by RUNES component model are realised in four different implementations:

- (i) a Java virtual machine-based implementation;
- (ii) a C/Unix-based implementation;
- (iii) an implementation based on tiny embedded devices running the Contiki OS; and
- (iv) an Erlang virtual machine-based implementation supported with a model-based software development framework (GME).



Rationale

An embedded system is a special-purpose computer system contained within a device that is expected to function without human intervention. The wide range of devices containing embedded systems inevitably leads to significant complexity in appropriately interconnecting and dynamically reconfiguring their software. New approaches in software design are needed to address the power, computation and communication bandwidth limitations but also security. RUNES provides an adaptive middleware platform and application development tools that allow programmers the flexibility to interact with the environment where necessary, whilst affording a level of abstraction that facilitates ease of application construction and use. The results of the project should be of interest to technologists, product managers, manufacturing managers and consultants working particularly in the following sectors:

- Healthcare
- Transport
- Utilities
- Industrial processes
- Security
- Oil & gas
- Food and agriculture
- Environmental

Contact

Project Manager:
András Tóth
Ericsson AB
Tel: +46 8 58530531

Email:
andras.toth@ericsson.com

Website:
www.ist-runes.org