Modelling the Configuration/Management API Middleware using Coloured Petri Nets

Paul Fleischer
Lars M. Kristensen

Department of Computer Science
University of Aarhus
DENMARK
{pf,lmkristensen}@daimi.au.dk
Background

- Joint research project involving:
  - TietoEnator IP Solutions, Denmark.
  - Coloured Petri Nets Group, University of Aarhus.

- Main theme of the research project:
  - Using Coloured Petri Nets and CPN Tools in the specification and implementation of protocol software.
Coloured Petri Nets

- Modelling language based on Petri Nets and the functional programming language CPN ML:

**Petri Nets:**
- concurrency
- control structures
- synchronisation
- communication
- resource sharing

**CPN ML (Standard ML):**
- data manipulation
- compact modelling

- Supported by CPN Tools: [www.daimi.au.dk/CPNTools](http://www.daimi.au.dk/CPNTools)
GAN Architecture

- This subproject is concerned with the **Generic Access Network (GAN)** architecture.
- GAN supports access to telephone network services via IP networks:

  ![Diagram of GAN architecture]

  - Mobile Station
  - Security Gateway
  - GAN Controller
  - IP network
  - Telephone network

- Currently being developed by the 3rd Generation Partnership Project (3GPP).
GAN Scenario at TietoEnator

- In earlier work we developed a Coloured Petri Nets model of the GAN scenario considered:
  - Step 1: DHCP
  - Step 2: Create ESP tunnel to Prov. SGW using Internet Key Exchange (IKE) protocol
  - Step 3: Acquire addresses of default SGW and GAN Controller
  - Step 4: Create ESP Tunnel
  - Step 5: Initiate GAN connection using Internet Key Exchange (IKE) protocol

- TietoEnator is now designing and implementing the protocol software for the mobile station.
The Configuration/Management API (CMAPI) middleware is used to control:

- The **IPSec protocol** module for encryption and authentication.
- The **IKE protocol** module for negotiation of IPSec parameters.
CMAPI Middleware

- Vendor-specific API and middleware for embedded systems implemented in C.
- CMAPI maintains a set of objects representing the system components that are being managed.

- Objects are structured in a parent-child hierarchy and created based on a set of CMAPI classes.
- CMAPI classes supports:
  - Construction and destruction of objects.
  - Set and get of parameters for the objects.
  - Actions on objects belonging to the class.
Demonstration of CPN model
Model Overview

- Mobile Station
  - Controller
    - CMAPI
      - Create CMAPI Object
  - IKE
    - IKEAction
      - CSAE Action
  - IPSec

20 controller submodules
Mobile Station Module

Controller

CMAPI Top

CMAPI

API

IKE_API

CMAPI Bottom

CMAPI_LOW

IPSEC

IKE

IKE_API

API

CMAPI

Controller
Controller Module
Controller Initialisation

Start

Call

Wait

Receive

Completed

Variables

CMAP OBJECT MAP

UNIVERSITY OF AARHUS
Department of Computer Science

LMK - PNTAP 2008
Controller Initialisation

```
(controller, 
Call(CreateObject(
cmaapi_class_ikev2_main_object, 
cmaapi_main,Ikev2MainParams ))))

1' (controller,Call(CreateObject((cmaapi_class_ikev2_main_object,2,[("vendor_id","Ericsson AB")]))))

Call

Wait

Receive

(controler, Return(Object obj))

("ikev2_main", obj)

Variables [I/O]

CMAPi_OBJECT_MAP

Completed

UNIT

Start

UNIT

()
Mobile Station

```
1` (controller, Call(CreateObject((cmap_i_class_ikev2_main_object, 2, ["vendor_id", "Ericsson AB"]))))
```
CMAPI Module

1. controller, Call(CreateObject((cmap, i_class, ikev2_main_object, 2, ["vendor_id", "Ericsson AB"])))

Objects

- Dispatch Object Create
- Create CMAPI Object
- Dispatch CMAPI Get
  - [#instance(obj) = (#1(get))] (obj, s, Call(Get get))
- Dispatch CMAPI Set
  - [#instance(obj) = (#1(set))] (obj, s, Call(Set set))
- Dispatch CMAPI Action
  - [#instance(obj) = (#1(act))] (obj, s, Call(Act act))
- Dispatch Object Destroy
  - [#instance(obj) = (#1(destroy_cmd))] (obj, s, Call(DestroyObject destroy_cmd))

Return

CMAPBOTTOM

CMAPI_TOP

CMAPI

CMAPI_OBJECT

I/O CMAPI

I/O CMAPI_LOW

I/O
Create CMAPI Object

```
{class=(#1(create_cmd)),
parent=(#2(create_cmd)),
instance=n}
```

```
(s,Call(CreateObject create_cmd))
```

```
1*(controller,Call(CreateObject((cmap
_i_class_ikev2_main_object,2,(["vendor_id","Ericsson AB"])))))
```
Create CMAPI Object

Create CMAPI Object

{class=(create_cmd), parent=(create_cmd), instance=n}

Create Object

CMAPI

CMAPI_TOP

{class=(create_cmd), parent=(create_cmd), instance=n}

Out

CMAPI_OBJECT

1`[{class=cmaapi_class_ikev2_main_object, parent=2, instance=3}, controller, Call(CreateObject((cmaapi_class_ikev2_main_object,2, ["vendor_id","Ericsson on AB"])))]

1`[{class=cmaapi_class_ikev2_main_object, parent=2, instance=3}]

1`[{class=cmaapi_class_ikev2_main_object, parent=2, instance=3}, controller, Call(CreateObject(create_cmd))]

n

n+1

In

Out

INT

CMAPI_LOW

Objects

Create Object

Instance Counter

1

1'4

1`
Mobile Station

```
1`({class=cmapi_class_ikev2_main_object,parent=2,instance=3},controller,Return(Object({class=cmapi_class_ikev2_main_object,parent=2,instance=3})))
```
Mobile Station

1 `(controller,Return(Object({class=cmapi_class_ikev2_main_object,parent=2,instance=3})))

Controller

CMAPI Top

CMAPI

CMAPI

CMAPI Bottom

CMAPI_LOW

IPSEC

IKE

Controller

API

IKE_API

IKE

IPSEC
Controller Initialisation

(Controller, Call(CreateObject(cmaapi_class_ikev2_main_object, cmaapi_main,Ikev2MainParams)))

(Controller, Return(Object(obj)))

Start

Call

Wait

Receive

Completed

Variables

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP

CMAP Pool

CMAP
Controller Initialisation

Start

Call

Wait

Receive

Completed

(variables)

Pool

CMAP

Controller Initialisation

Call(CreateObject(
    cmapi_class_ikev2_main_object,
    cmapi_main,Ikev2MainParams )))

(controller, Return(Object obj))

Call("

("ikev2_main", obj)

1"{"ikev2_main","class=cmapi_class_ikev2_main_object,parent=2,instance =3"})

"ikev2_main", {class=cmapi_class_ikev2_main_object,parent=2,instance =3})

Start

UNIT

UNIT

UNIT

UNIT
Validation

- CPN model validated using simulation and state space exploration.

- State space: 220 nodes and 360 arcs.

- Key properties established:
  - The controller always eventually enters the desired running state(s).
  - The controller, CMAPI layer, and IKE module are consistently configured.
  - Only a finite number of CMAPI objects are created.
Conclusions and Future Work

- Initial CPN model of the controller software and the CMAPI middleware layer.
- Developed and reviewed in close cooperation with engineers at TietoEnator.

Future work:
- Extend model to include IPSec module management.
- Complete the simulation and state space analysis for validation of the controller design.
- Code generation of the controller software based on the constructed CPN model.