Design and Verification of Real-Time and Communicating Software

Michel Diaz, François Vernadat
LAAS-CNRS

LAAS-CNRS
Octobre 2008

40ième Anniversaire du LAAS
I

INTRODUCTION
Design

• **Define** properties, scenarios or services

• **Design Phases**
  - Spec, Val, Impl, Test

• **Hierarchy of Design Steps**
  - Mechanisms, components, modules, levels, etc

• **Using models**
  - Physical and Logical models
  - Software and Hierarchical models
System Models

• **Basic Models**
  – Extended (Timed, etc) **Petri Nets**
  – Extended **State Machines**
  – Extended **Process Algebra**

• **Description Techniques**
  – Formal: Estelle, SDL, Lotos (FDTs for **Protocols**)
    • Estelle and SDL, Extended State Machine oriented
    • Lotos, Extended Process Algebra oriented
  – Semi-Formal:
    • **UML** (the pioneer)
    • AADL, SysML, UML2 (Object oriented, including SDL)
From Design to Verification

• For Each phase/level, as soon as possible, using a global model/representation of the system, Verify or Simulate its behaviour

• Verification based on Modal Logic
  – Because values of propositions evolve dynamically
  – Axiomatic proofs not automatic and difficult
  – Semantic proofs from Kripke Structure

• Kripke Structure (set of connected Worlds)
  – Primitive Predicate symbols (p, q, r, …)
  – Interpretations for p, ¬p, and, or, .. for a world \( W_i \)
  – Modalities from a set of worlds connected by a relation \( R \)
Semantics in Modal Logic

• The worlds are the system states
• R is accessibility relation between worlds (global behaviour)
• Technical approaches and tools based on the graph (whatever defined) by (Linear or) Branching Modal Logic
From Telecommunications to Embedded and Internet Systems

• **Embedded Systems**
  – Architectures
  – Behaviour
  – Properties (functional and non functional (time, energy, ...))
  – Models, Verification, Evaluation

• **Internet Systems**
  – Architectures
  – Behaviour and Performances
  – (Minimal) Acceptable Non Optimal design : Best Effort
  – Simulations (mainly of implementations)
II
EMBEDDED SYSTEMS
Embbeded systems based on

- System Specification and Requirements
- Design steps
  - Technologies to mechanisms
  - to equipments to architectures
- Models
- Verification
  - Full behaviour and Properties
  - Automatic by Tools

- Design supports
  - Formal, Verified Designs, *e.g.* TOPCASED
TOPCASED Project Overview

- Open Source system development environment

- Implementing an integrated model-based development process
  - from system specification
  - to the final product, including formal verification.

- Reduce development costs by optimised process and tools

- Integrate MDE and formal verification by
  - Meta-Modeling, Process modeling
  - Model Verification, simulation, static analysis
  - Model Transformations
TOPCASED

Analyses or Design Model

Simulation

Verification loop

Formal checking

Transformation

Source or Test Code

Documentation

Configuration, Change and Requirement management tools communication
Intermediate Language: Fiacre

(Meta)-modeller

Editors

Model Transformations

Intermediate language

Compilers

Model-Checkers

Simulators

Modelling languages

AADL  SDL  UML  SYSML  ...

ATL, Kermeta

Fiacre

Translation

CADP  Tina  ...

PDL
type request is union get_sum, get_value of index end ...

process ATM [req : in request, resp : out nat] is
states ready, send_sum, send_value
var c : request, i : index, sum : nat, val : data := [6, 2, 7, 9]
init to ready

from ready
  req ?c
  case c of
    get_sum → to send_sum
    | get_value (i) → to send_value
  end
from send_sum
  sum, i := 0, 0;
  while i < 3 do
    sum, i := sum + val[i], i + 1
  end;
  sum := sum + val[i];
  resp !sum;
  to ready
from send_value
  resp !val[i];
  to ready
Verification by PN Based models and TINA

• **Including Time**
  – Time Petri Nets (intervals on transitions)
  – Analysis based on State Classes (symbolic, DBMs)

• **And Priorities**

• **And Suspension/Resumaption**
  – Time Petri Nets + Stopwatches
  – State Classes + Over-approximations

• **And Data**
  – Time Transition Systems (TS + Time) & High Level Descript
TINA Tool box (Time PN Analyser)

**nd**: Editor
- Import/export
- Simulator

**tina**: State spaces

**selt**: LTL model-checker

**struct**: Structural Analysis

**plan**: Path Analysis

**ktzio**: LTS Conversions

**ndrio**: Net Conversions
III
INTERNET SYSTEMS
Internet Systems

• **Two approaches**
  – 1. from architecture to layers
  – 1. from layers to entities \(^1\)
  – 2. from mechanisms to protocols
  – 2. From protocols to entities \(^2\)

• **Design efforts**
  – From Best-Effort to QoS Internet &
  – to Guaranteed QoS, *e.g.* EuQoS
QoS Internet

- From QoS Applications
- How to derive networks and architectures
- satisfying QoS Bandwidth and Time requirements
A Best-Effort MULTIMEDIA Architecture

Applications (code)
Application Framing and (QoS?) Control

- Losses detection
- Flow Control
- Buffering
- Losses recovery
- Congestion Control
- Synchronization

RTP
UDP

Best effort Network (IP - BGP)
The 3 QoS internet Approaches

1. Network Overprovisionning

2. Optimised Best-Effort mechanisms, protocols and architectures

3. New Internet Architectures to guarantee the QoS
QoS Optimisation

• **Start from Best-Effort Internet**
  – without modifying the architecture principles
  – using resources/bandwidth available
  – analysing and improving present solutions

• **Modify mechanisms and protocols**
  – modify applications (adaptativity, new codecs,…)
  – optimise architecture (proxys,…)
  – define new protocols (Transport Layer: DCCP,…)

• **But still Best-effort (No guarantee)**
QoS (hard) Guarantee

New requirements

• **Master** the Internet
• **Be as General and Open as** the present Internet
• **Propose** new mechanisms, protocols, architectures
• **Handle** sessions and resources

Main problems

• **Resulting Complexity ?**
• **Difficulty of Deployment wrt the present internet ?**
Vertical (Applis-to-Networks) and Horizontal (Host-to-Host) problems

QoS requirements

Applications
- bandwidth
- Time constraints (delay, jitter)
- Synchronization
- Partial reliability and order

Mechanisms and protocols in the middleware

Multi-protocols

QoS-oriented Transport protocols

IntServ (RSVP) IP Serv DiffServ

Infrastructure capability

Access network

Access network

??? ???

Access network

Access network

Access network
A lot of work done (for QoS)

- Many mechanisms and protocols
- Many partial architectures

But HOW to INTEGRATE

- in a globally coherent
- and easy to deploy way
- from User to User:
  - Performing mechanisms
  - Their efficient composition in needed protocols
  - ALL protocols, e.g. data and services
EuQoS : Design Meta-Rules

- **Design the complete architecture**
  Mechanisms designed isolated from global context have a low probability to lead to satisfactory solution

- **End2End identical solutions cannot work**
  given the complex and geographical topology, the approach must handle diversity

- **Only key Signalling/Interfaces to be defined**
  Freedom to be given to designers in each technology to develop their most efficient solutions

=> Virtualize and Abstract Domains
Abstract Virtual Network Layer

Resource Managers (RM)

RM1  RMi  RMj  RMk  RM2

Application Layer

Application QoS-based signaling

Network Technology Independent sub-layer

IMPORTANCE:
1. OF SIGNALLING
2. OF ABSTRACT MODELS
Abstract Models in RMs
Ex: F (Border Routers)
such that: $P_{QoS}$ on $Ta(Ma) \Rightarrow P_{QoS}$ on $Tr(Mr)$

Model Ma

Abstract Topology $Ta$

Model Mr

Topology $Tr$

$P_{QoS}$
Application Layer

Abstract Virtual Network Layer

Application QoS-based end-to-end signaling

Network technology Independent sub-layer

Resource Managers

Network Technology Dependent s-l

Ressource Allocators

Access Network 1

QoS Domain i

QoS Domain j

QoS Domain k

Access Network 2
Main Design Steps

1. **Independence of:**
   - Applications wrt Virtual networks wrt
   - Virtual networks wrt Technologies
   - Signaling wrt Data Plane

2. **Integration of Applis with**
   - QoS Invocation (Admission Control)
   - Defined full Architecture
   - Linked to main present solutions
   - Linked to scalability
   - **QoS Network layer:** CoS (Classes of Services)
   - QoS Signalling
   - QoS Transport layers
QoS Network Layer: Classes of Services

<table>
<thead>
<tr>
<th>Classes of Service</th>
<th>EQ-CoS CoSs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT</td>
<td>Maximum Bandwidth = G</td>
</tr>
<tr>
<td>NRT</td>
<td>Minimum Bandwidth = g</td>
</tr>
<tr>
<td>BE</td>
<td>No guarantee</td>
</tr>
</tbody>
</table>
E2E CoSs – aggregated QoS and CoSs for EuQoS

- EUQOS Applications
  - VoIP
  - VTC
  - VoD
  - Data Transfer

- End-to-end CoSs
  - Telephony
  - RT interactive
  - MM streaming
  - High throughput data

- CoS on Access Network
- CoS in Inter-domain
- CoS on Access Network

- End-to-end CoSs
- Telephony
- RT interactive
- MM streaming
- High throughput data

- EUQOS Applications
  - VoIP
  - VTC
  - VoD
  - Data Transfer

All Networks
QoS EuQoS SIGNALING (EQ)

- Appli-to-Appli coding: EQ-SDP
- Appli-to-Appli QoS: EQ-SIP
- Appli-to-Virtual network: EQ-QoD
- Virtual Network CoS: EQ-NSIS
- Virtual-to-Real networks: COPS
- 3 classes QoS Routing: EQ-BGP
- End-to-End path: EQ-path
- Telcos MPLS integration: EQ-PCE
QoS Optimised Transport Layer: ETP

For Data

UDP

SCTP

TCP

ETP

ORDER

DCCP
<table>
<thead>
<tr>
<th>Network Classes of Service</th>
<th>Application profile</th>
<th>Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT</td>
<td>ETP=UDP[RC]</td>
<td>ETP[EC]</td>
</tr>
<tr>
<td>NRT</td>
<td>ETP[gTFRC]</td>
<td>ETP[gTFRC+EC]</td>
</tr>
<tr>
<td>BE</td>
<td>ETP[TFRC+DT]</td>
<td>ETP[TFRC+DT+EC]</td>
</tr>
</tbody>
</table>

**Error tolerant e.g. VoD**

**Error intolerant e.g. file transf**
The EQ-Path including domains (BGP-based) and sets of domains (MPLS-PCE)
IV
CONCLUSION
The Future of Embedded systems

• Types of systems
  – From SW (timed models) to SW/HW (hybrid systems)
  – Systems of Systems
  – Mobile Systems
  – Distributed and Networked Systems

• Properties, Algorithms and Tools
  – Quantitative analysis
    • Schedulability analysis, consumption
  – High level constructs integrating formal models
  – Scalability: Managing Combinatorial Explosion
    • Compositional verification
    • Parallel model-checking
    • Abstractions (e.g. preserving properties), etc
The Future of Internet systems

- Full mobility
- Network of the future (e.g. GENI, FIRE)
- Internet Virtualisation
  - Virtualised routers able to run in parallel a set of different protocols
- Application-aware networking
- Sensor networks and ad-hoc networks
- Internet of the Things
  - => importance of the sensor & things (values, etc)
- Real-Time internet
Integration

To Go from Embedded system to a (given) sub-set of the Future Internet

- Include some Sensors and Things, with mobility
- Define Real-Time protocols from Applications
- Integrate Multilayering and Composability
- Develop Easily Verifiable Methodology

(Extending adequate methods and tools)
MERCI