Potential new case studies for behavioural types:
Switching software systems

Tihana Galinac Grbac
Faculty of Engineering
University of Rijeka
Croatia
Summary

- Introduction
- Switching software
- System requirements
- Related work on formalisms
- Concrete protocol examples
- Cases study: system description
- Call flows
- Problems: previous experiences
- Switching software perspective
What is switching software?

Switching software in telecommunication network

Picture taken from: http://commons.wikimedia.org/wiki/File:WAC_telephone_operators_operate_the_Victory_switchboard_during_the_Potsdam_Conference_in_their_headquarters_in...-_NARA-_199007.jpg
Switching software

- Telecommunication exchanges are developed more then 50 years
- Main concepts are space division and time division
- Numerous formal approaches were developed
- Still we face with number of failures
System requirements
System has to provide:

• Parallel execution of multiple different requirements, for number of users
  – e.g. Systems implementing MSC logic has to cope with more than milion requests in parallel,
  – provide number of different ‘standard’ protocol interactions
• high availability for its users
  – If certain malfunction happen the peers has to be timely informed, and all related resources properly released, avoid congestion situations
• properly dimensioned – aviod load
• response by the required time
  – Real time system, a system with a real-time constraints
• Interoperable with other vendors equipment
• Inside logic has to provide external protocol compliance
System requirements
System has to provide:

• Easy to maintain
  – system structured into number of logical functions
  – Well defined and separated logical functions
  – Easy to trace system dynamics
  – Easy transformed from object code back to original code
Switching software systems: Programming languages examples

• PLEX: used in Ericsson AXE telephone exchange
  – Concept of program segments: data encapsulated by set of procedures (modules) that could access data, other modules can not
  – Software is living modular system and basic requirements are flexibility, manageability, ease of modification, ease of handling, etc.

• CHILL: Used in 1240 ITT telephone exchange
  – Has to support more than 100 000 telephone calls simultaneously, fully distributed control
  – Concept of ‘finite message machines’
    • Predefined set of input messages and each produces a finite set of replies
    • No FMM has direct access to memory of another FMM – all communication through messaging
    • FMM may reside on the same or different machines – message handler implements routing functionality among the machines for each messages

• Erlang
  – Reliable system but in presence of errors

Tihana Galinac Grbac
University of Rijeka, Croatia
Switching software systems 1: Application of formal methods to the verification of communication protocols

- the most commonly used methods for ensuring the correctness of a system are simulation and testing
- exhaustive for any reasonably complex system is impossible
- Errors can sometimes occur only for specific execution sequences which are difficult if not impossible to reproduce or debug, making an exhaustive analysis necessary
- Knowledge that the design logic of a system is correct help us to avoid critical system faults
- the errors during implementation are less likely to be fundamentally logical errors that are very difficult to understand and solve
Switching software systems 2:
ITU-T Standardisation, Z series

- Unified Modeling Language UML diagrams, Message Sequence Chart MSC
- Specification and Description Language SDL (ITU-T)
- for the description of communication protocols including concurrent and real-time aspects
- System is represented as set of blocks and processes,
- Processes interact asynchronously via signals that are placed into and consumed from queues.
- The communication structure is given by signal routes that connect individual system components.
- Used as automatic verification techniques like model checking
- high-level languages and formalisms particularly made for real-time and communication systems, CHILL
Concrete protocol examples

• Signalling network and protocols in cellular telecomunication network
Network communication architecture

- Networks are defined and modeled at different abstraction levels

**Open Systems Interconnection Model**

- Application Plane
- Control Plane
- Data /Resource Plane

Network architecture
Case study

• Signalling network in mobile communication network
• We can consider as high priority data traffic network
• Developed according 3GPP standardisation body regulations
Evolution of telecommunication network

• 3rd Generation Partnership Project
  – Standards for cellular telecommunications network technologies
  – http://www.3gpp.org/
Signalling network evolution
Phase 1: Introduction of ATM transport and new BICC protocol

COST Action 1201 Behavioural Types for Reliable Large-Scale Software Systems
WG/MC Meeting: Sunday 31st August 2014 in Rome

Tihana Galinac Grbac
University of Rijeka, Croatia
Signalling network evolution
Phase 2: Splitting of network architecture - new GCP protocol

COST Action 1201 Behavioural Types for Reliable Large-Scale Software Systems
WG/MC Meeting: Sunday 31st August 2014 in Rome

Tihana Galinac Grbac
University of Rijeka, Croatia
Signalling network evolution

Phase 3: Migration to IP network

COST Action 1201 Behavioural Types for Reliable Large-Scale Software Systems
WG/MC Meeting: Sunday 31st August 2014 in Rome

Tihana Galinac Grbac
University of Rijeka, Croatia
Protocol example

- Interaction between horizontal and vertical protocol
System

- AXE based MSC node
- PLEX signals
- Several million lines of code
- Divided into more than 1000 modules
- More than 1 million of simultaneous requests
- Priority levels
- Resources
- Real time with time constraints
Call set up

Galenac Grbac
University of Rijeka, Croatia
Call flows

• Example: Session Initiation Protocol
Problems (1)

• Several levels of abstractions
• How to define right structure - and secure type safety between internal and external interfaces
• The most critical issues in the maintainance – from the aspect ‘How hard is to locate the fault’
• Interactions between system functions and network functions
Problems (2)

- Protocol interactions
  - E.g. Node recovery, congestion control, etc.
  - System internal optimisations aiming to increase the system performances (e.g. Dual seizure – shortcut in internal communication)
- Partially executed system procedures may affect inconsistencies
- Order of release procedure is important (Control process)
- System configuration change that may take longer time because of lower priority
- Shared resources and correct release procedures, relinking of blocks, traffic interference
- Message multiplication, congestion and time constraints
- Timers in the system and their interaction (are some timers long enough)
- Priorities of error cases
Switching software perspective

- Connectivity software, implemented in software switch
  - include call agents, call servers and media gateway controllers
- Becomes to be increasingly important
  - Internet of Things – anything can communicate with anything and anybody
  - M2M interaction
  - Software Defined Network – software can reconfigure itself at runtime
  - Open Source community - Everybody could contribute to distributed application development
- Consequences of failure are higher then ever
- Problems of dynamic restructuring of system design becomes the most important mechanism