Co-Design Patterns for Embedded Network Management

Dominique Dudkowsk
NEC Laboratories Europe
Heidelberg, Germany
dudkowsk@nw.neclab.eu
Agenda

- Motivation
- Embedded Network Management
- Co-Design Patterns
- Fault Management Scenario
- Benefit Evaluation
Motivation (1)

Current network and service infrastructures are
- complex and heterogeneous
- large-scale deployments
- provided by many stakeholders
- deployed dynamically

Operations and **management** more challenging than ever!

Some principles to tackle the challenges of network management
- Structuring: e.g. modularity, layering, hierarchies
- Various forms of interaction: e.g. cross-layering

Another successful approach to tackle complexity and scale
- **Design patterns**: e.g. singleton, facade, iterator, and many more ...
Motivation (2)

Lacking support for **structured design of embedded management functions**

- Current practices / traditional approaches (e.g. SNMP, ...)
  - Management functions **outside** the network
  - Network elements are “dumb”
  - Management tasks are performed per device
  - Effective for small networks
- Not adequate for emerging technologies
  - Both large-scale and dynamic, autonomic
- Limitations of traditional approaches
  - Poor scaling, long reaction times
  - Frequent interactions needed between management and managed system

→ New approach needed!
Embedded Network Management (1)

- Clean slate approach
  - Abandon traditional assumptions
  - Placement of management functions, interaction **patterns**

- Enabling concepts
  - Decentralization, embedding of functionality, autonomy
  - Delegate task to a self-organizing management plane

- Goal: effective, scalable management with business value
Embedded Network Management (2)

4WARD: In-Network Management (INM)

- functional component
  - contains embedded management functions
  - management co-located with service logic

Node View
Embedded Network Management (3)

4WARD: In-Network Management (INM)

- distributed control loop
  - formed by collaborating functional components

Network View
Co-Design Patterns (1)

- **Co-location** of service and management functions captured by the embedded nature of in-network management → *structural* property

  - *Functional*: pertains to *design*

- **Co-design** of management and service functions
  - Mutual exploitation of service and management functions to achieve complex, distributed management tasks within control loops

- **Co-design patterns**
  - Supporting in the structured design and exploitation of synergies between management and service realm
  - Set of structural blueprints (just like design patterns) of how to construct management functions by combining knowledge and functions of both management and service realm
  - Facilitate function reuse, simplify management functions, and ultimately lead to an increase in overall system performance
Co-Design Patterns (2)

Paper objective

- Devise a first nonexhaustive set of (the more obvious) design patterns for embedded network management
- Initiate discussion on the benefits and potential of such patterns

Let's look at a few of them ...

Informed Handover

- Handover makes explicit that control is transferred to management side
- Clarifies functional distinction between service and management side
- Synchronous or asynchronous
- Informed: attach information ("hint")
- Example: security exception to be handled within certain time constraints → security management capability
Co-Design Patterns (3)

**Predicate**
- Defined way to evaluate a condition (hence, predicate) cooperatively
- Typical application to sharing of knowledge and function
- Predicate knowledge in service logic
- Predicate evaluation in mgmt logic
- Example: fault management

**Control tunnel**
- Enables the transfer of some state through the mgmt realm
- State semantics only known and relevant to service side
- Example: utilization of robust mechanisms provided by management realm
Fault Management Scenario (1)

- Data storage in mobile ad-hoc network
- Normal operation: single network partition → no faults
- Data migration between lightweight data servers N1 and N2 succeeds

- data server: operational
- data server: designated
- client node
- node storing server record of N1
- node storing server record of N2
- storage reference point
- server advertisement message
- data migration message
- redundancy detection message
- client query message
Fault Management Scenario (2)

- **Network partitioning** occurs
- In some cases, migration failure can lead to redundant data servers
- Result: N1 and N2 in different partitions → **server redundancy** (fault)

*Diagram:
- Node N1
- Node N2
- Node N3
- Data servers: operational, designated
- Client node
- Node storing server record of N1
- Node storing server record of N2
- Storage reference point
- Server advertisement message
- Data migration message
- Redundancy detection message
- Client query message
Fault Management Scenario (3)

Detection of server redundancy at N3
It is possible to detect redundancy by server advertisements implicitly

Redundancy notification sent to e.g. N1, which handles fault

- data server: operational
- data server: designated
- client node
- node storing server record of N1
- node storing server record of N2
- storage reference point
- server advertisement message
- data migration message
- redundancy detection message
- client query message
Fault Management Scenario (4)

- Applicable co-design pattern: **predicate**
- Configuration of received messages understood by service
- Evaluation by mgmt side

Diagram:
- N1, N2, N3 nodes
- Management capability: predicate evaluation
- True and false synchronous call
- Further processing
- Incoming messages
Fault Management Scenario (5)

Applicable co-design pattern: **control tunnel**

Detection is tunnelled to one of the servers

Fault handling by parameterized data migration mechanism

1. fault detection
2. fault indication
3. fault recovery

control information: target algorithm and algorithm parameterization
Benefit Evaluation (1)

- Analytical model and comparison of two approaches

**Co-designed solution** according to previous slides

**Non-co-designed solution**
- Single management station polls complete network regularly
- Local redundancy evaluation of collected information
- Fault indication to relevant servers

**Performance metrics**
- **Mean fault recovery time:** Time from fault detection to its resolution
- **Communication cost:** Total cost required for execution of single fault recovery
Benefit Evaluation (3)

- Communication cost [packets/second]
- Checking interval [seconds]

**Graph Details:**
- **Lines:**
  - 6 migration packets
  - 60 migration packets
  - 600 migration packets
- **Slopes:**
  - Negative slope: non-co-designed
  - Constant values: co-designed
Conclusion

Traditionally, design patterns have aided significantly in the design of complex software systems.

In network management, design methodologies have been used recently to support future Internet architectures.

This work: co-design patterns for embedded management
- Identification of a first set of patterns

Future work
- Naturally: extension to other types of patterns
- Gain experience in the value of the use of such patterns

Co-design patterns may contribute to supporting simplicity, reusability, and the distribution of management functions.
Empowered by Innovation

Dominique Dudkowski
NEC Laboratories Europe
Heidelberg, Germany
dudkowski@nw.neclab.eu