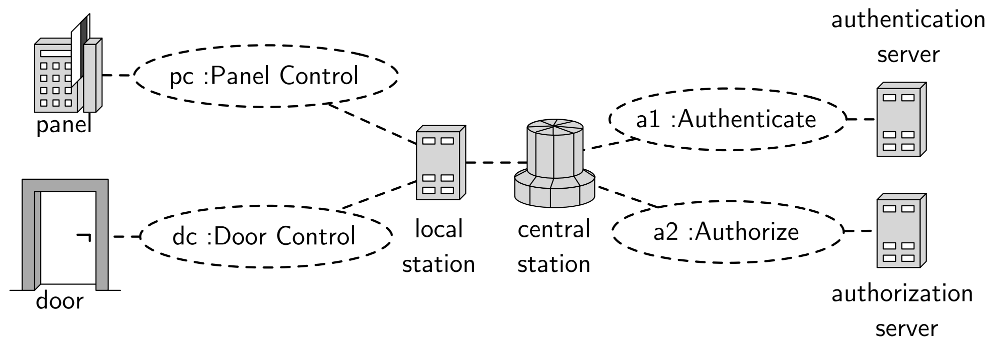
Towards a Model-Driven Method for Reliable Applications:

From Ideal to Realistic Transmission Semantics

Vidar Slåtten, Frank Alexander Kraemer and Peter Herrmann
Department of Telematics
Norwegian University of Science and Technology (NTNU), N-7491 Trondheim, Norway
{vidarsl, kraemer, herrmann}@item.ntnu.no

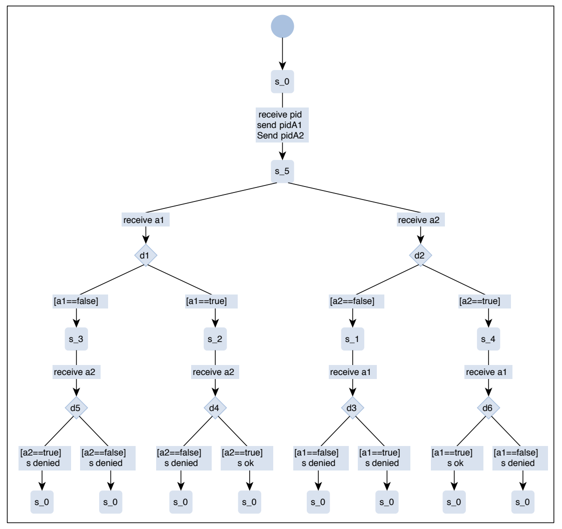
NTNU
Norwegian University of Science and Technology

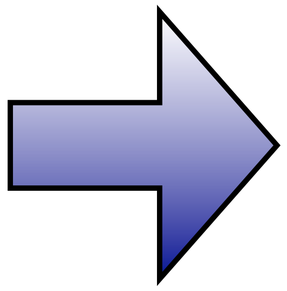
Overview – Problem

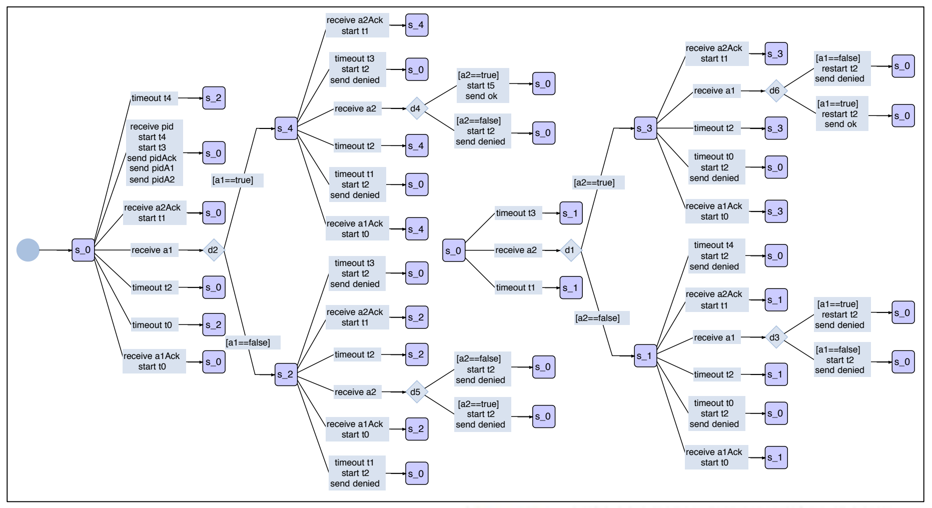


Developing distributed, reactive applications is hard

Developing **reliable**, distributed, reactive applications is even harder!

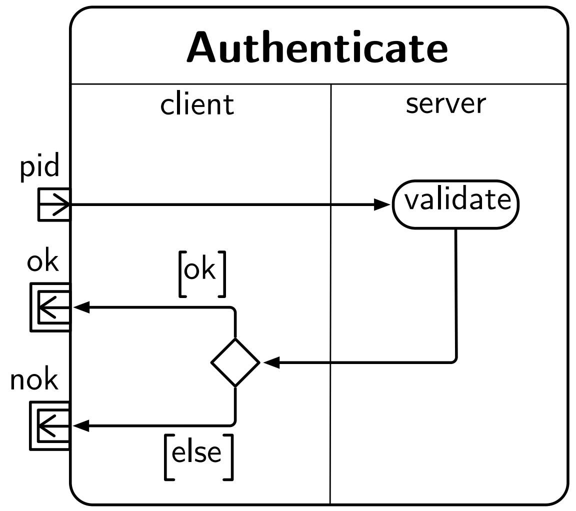




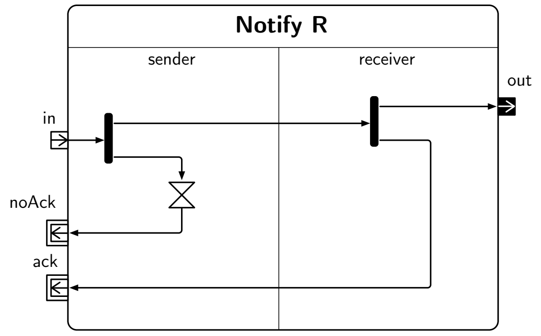


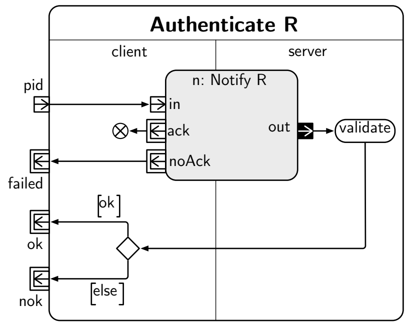
NTNU
Norwegian University of Science and Technology

Overview – Solution

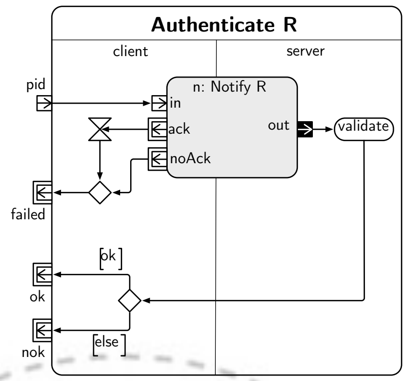


- Decompose application into building blocks encapsulating distribution
- Allow for an idealized specification (no operational faults) to be developed first





- Add encapsulated fault-tolerance mechanisms in a second step

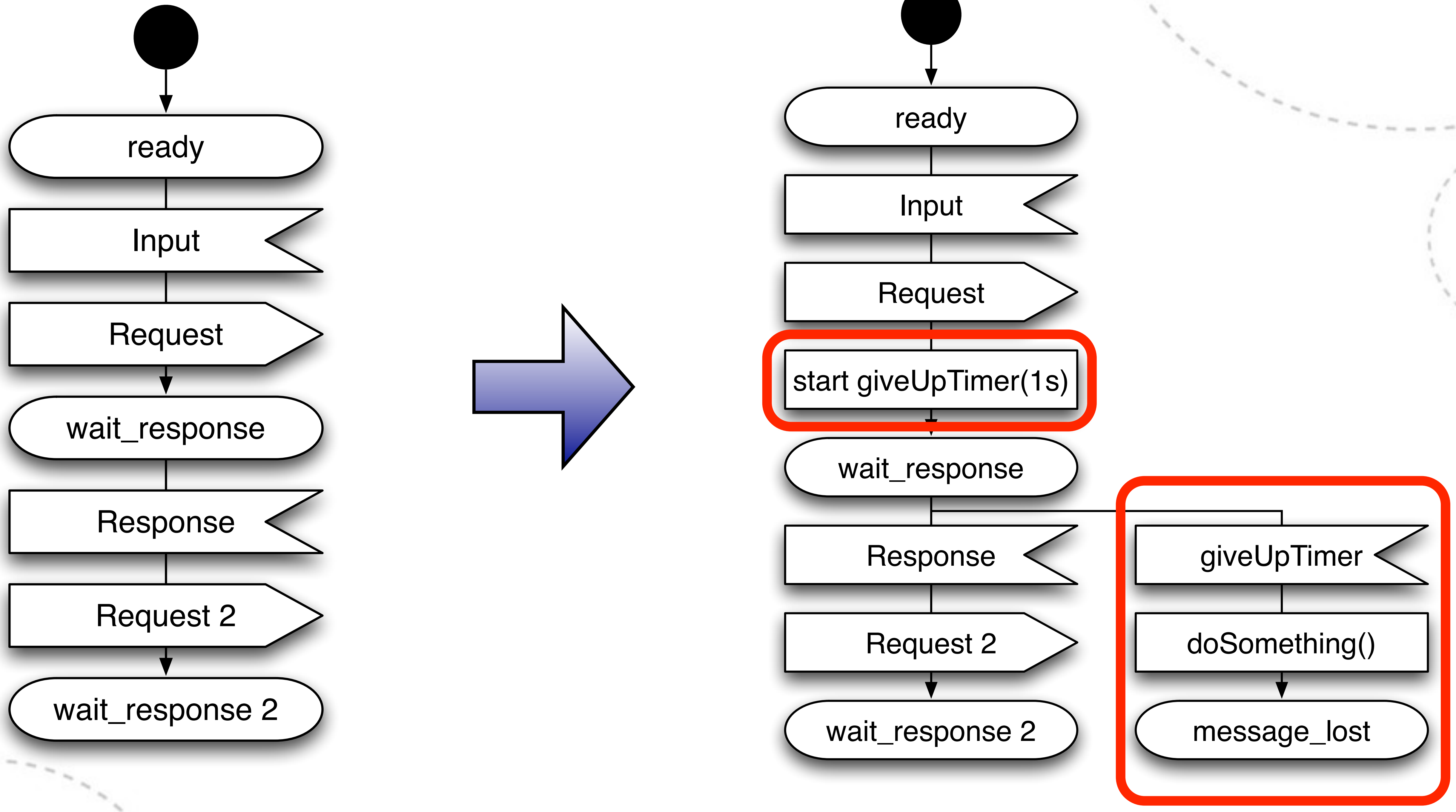


- Use tools for fault removal at every step

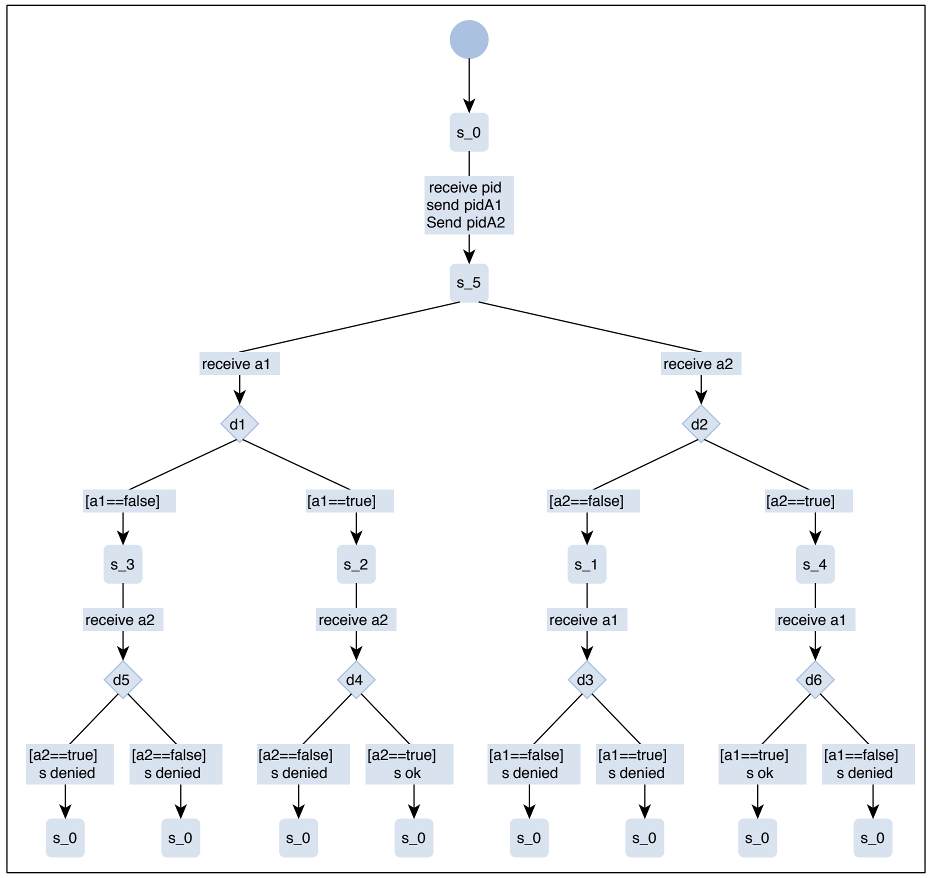
NTNU
Norwegian University of Science and Technology

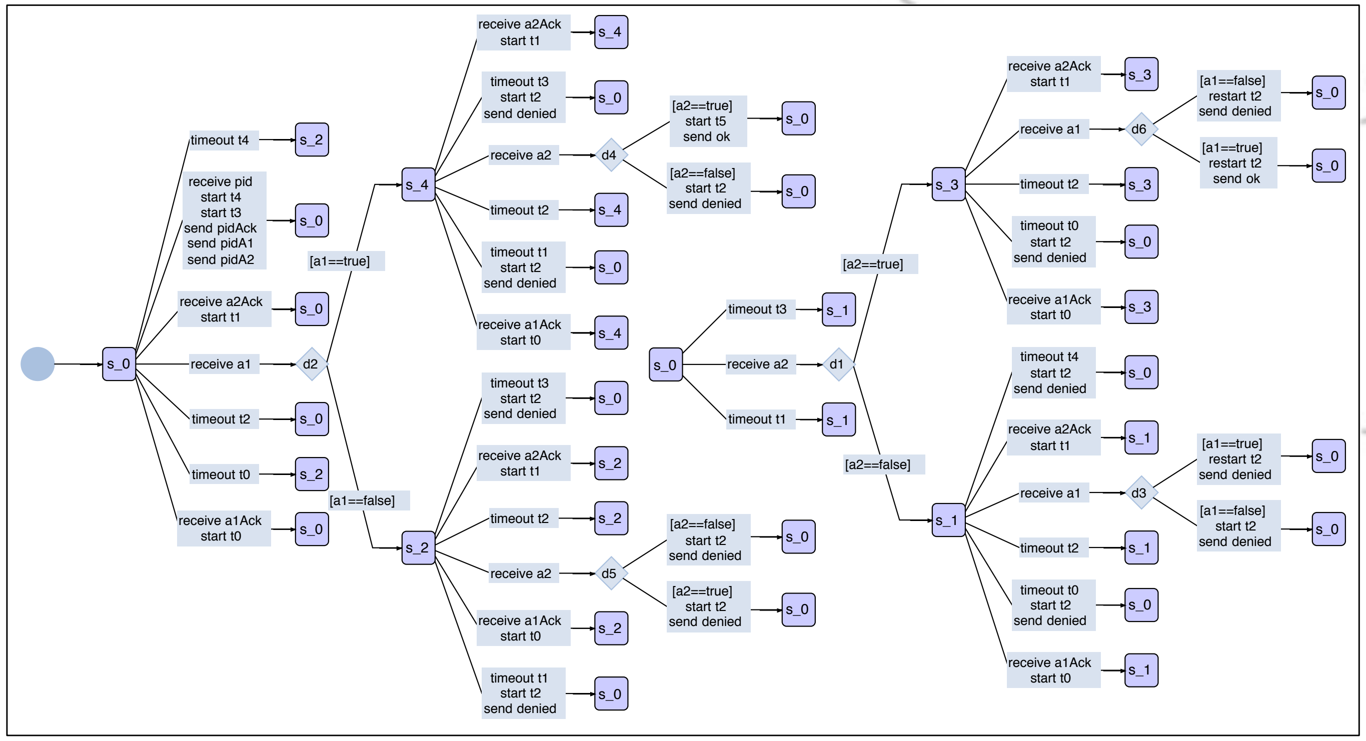
Scope: Unreliable channels

Add timeouts to detect possible message loss



Message loss detection – the size problem

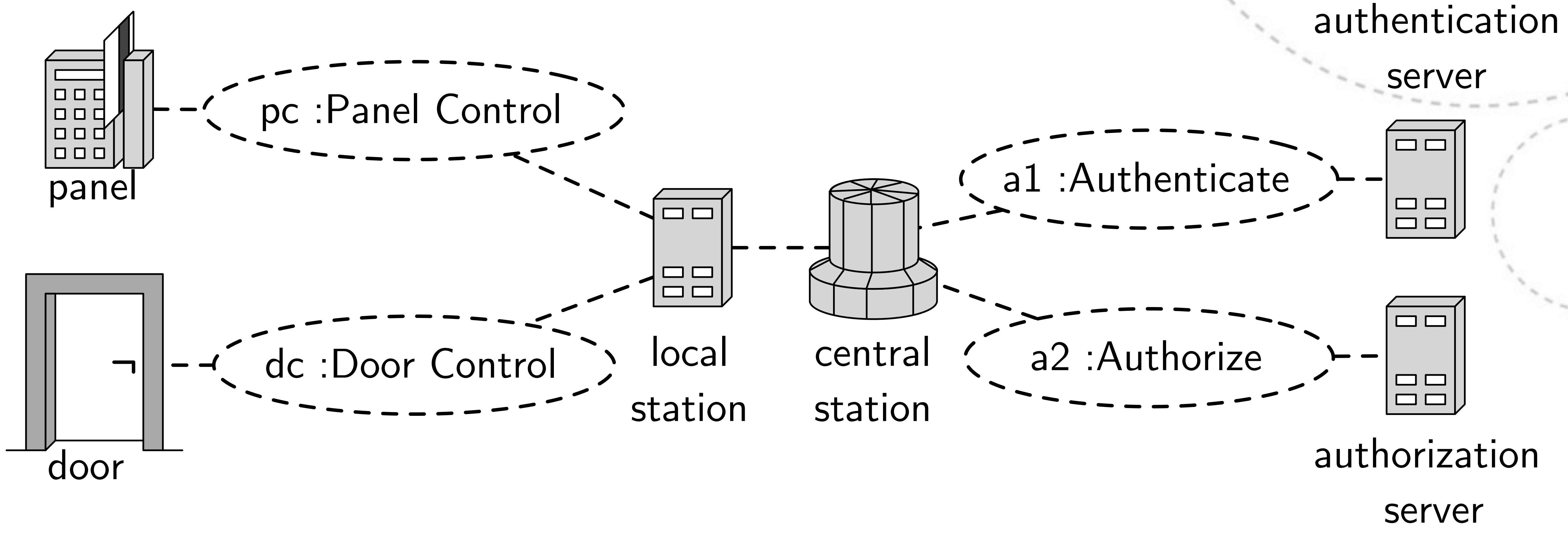




14 transitions

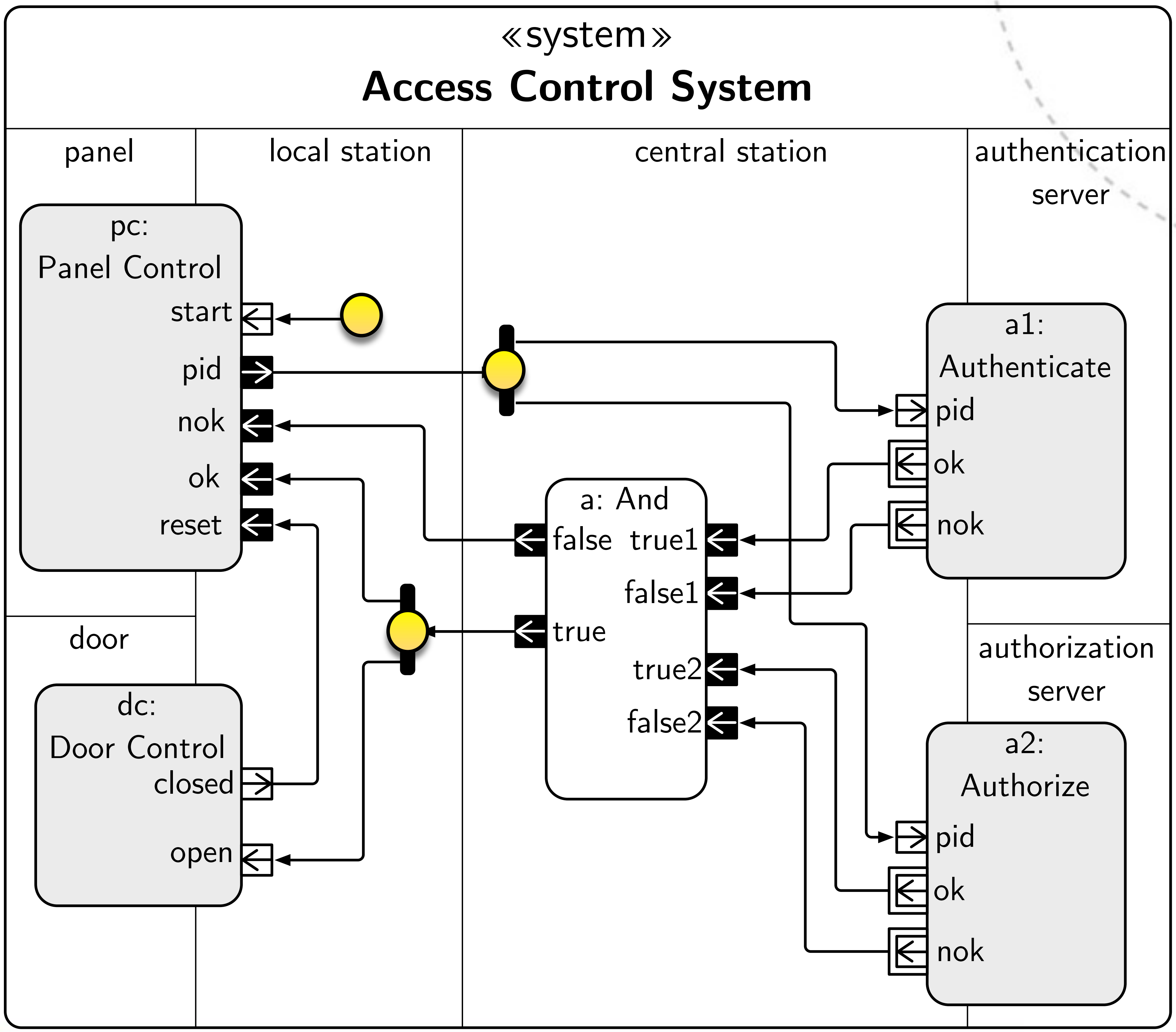
40 transitions

Example – ACS

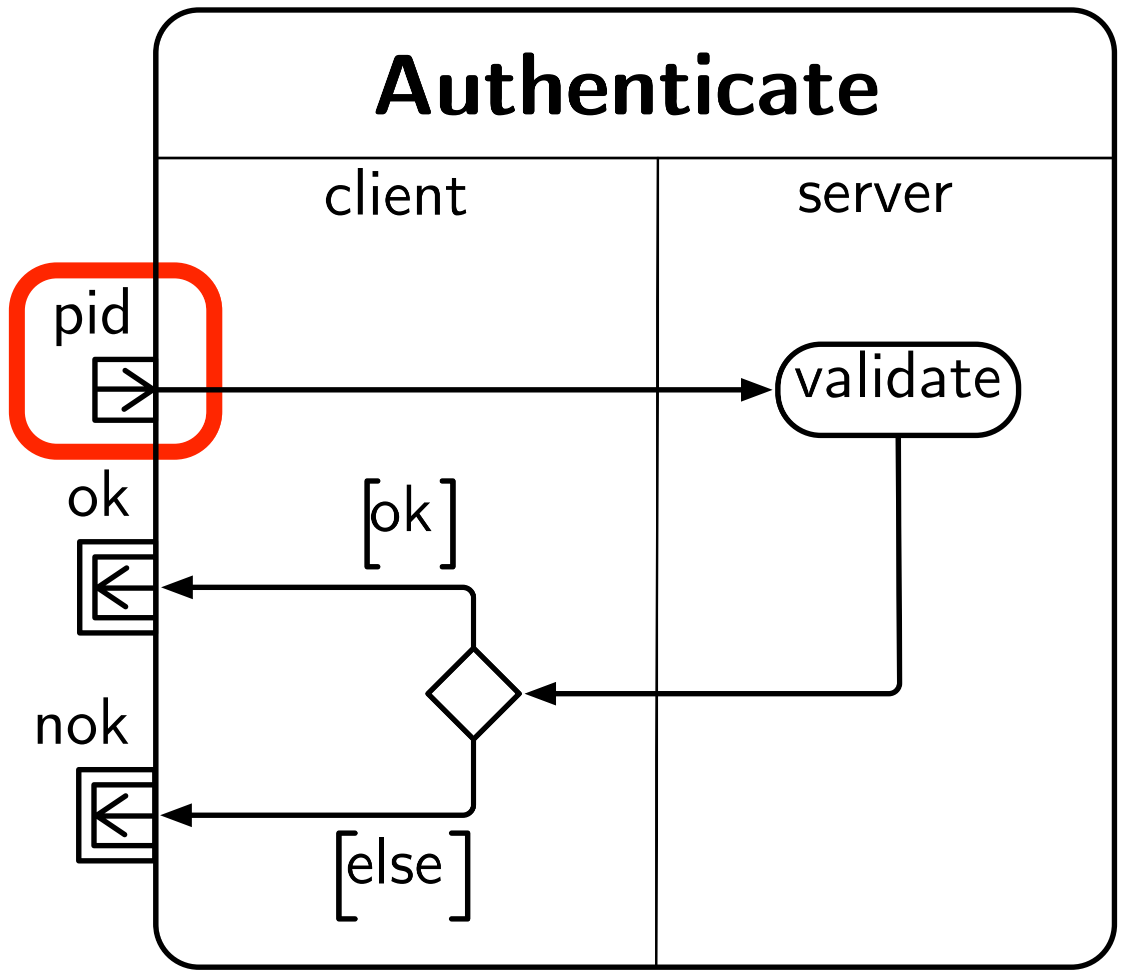


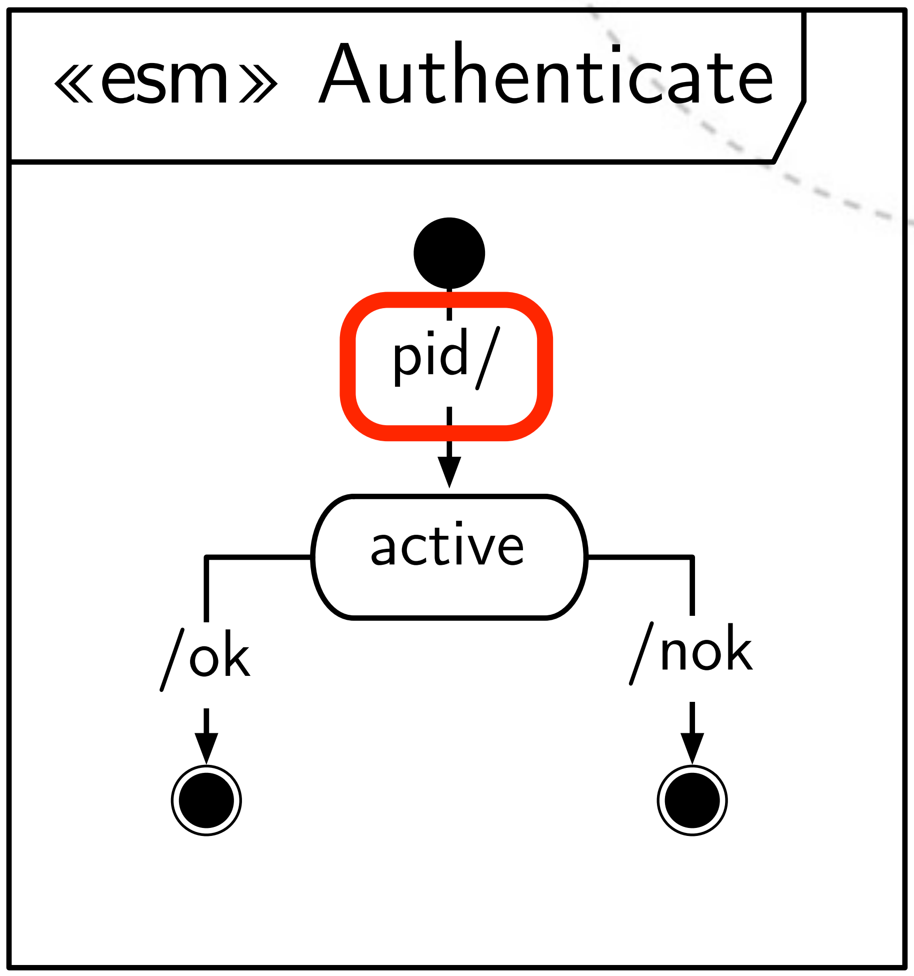
NTNU
Norwegian University of Science and Technology

Example – ACS


«system»
Access Control System
panel
local station
central station
authentication server
pc: Panel Control
start
pid
nok
ok
reset
a1: Authenticate
pid
ok
nok
a: And
false
true1
false1
true
true2
false2
door
dc: Door Control
closed
open
authorization server
a2: Authorize
pid
ok
nok

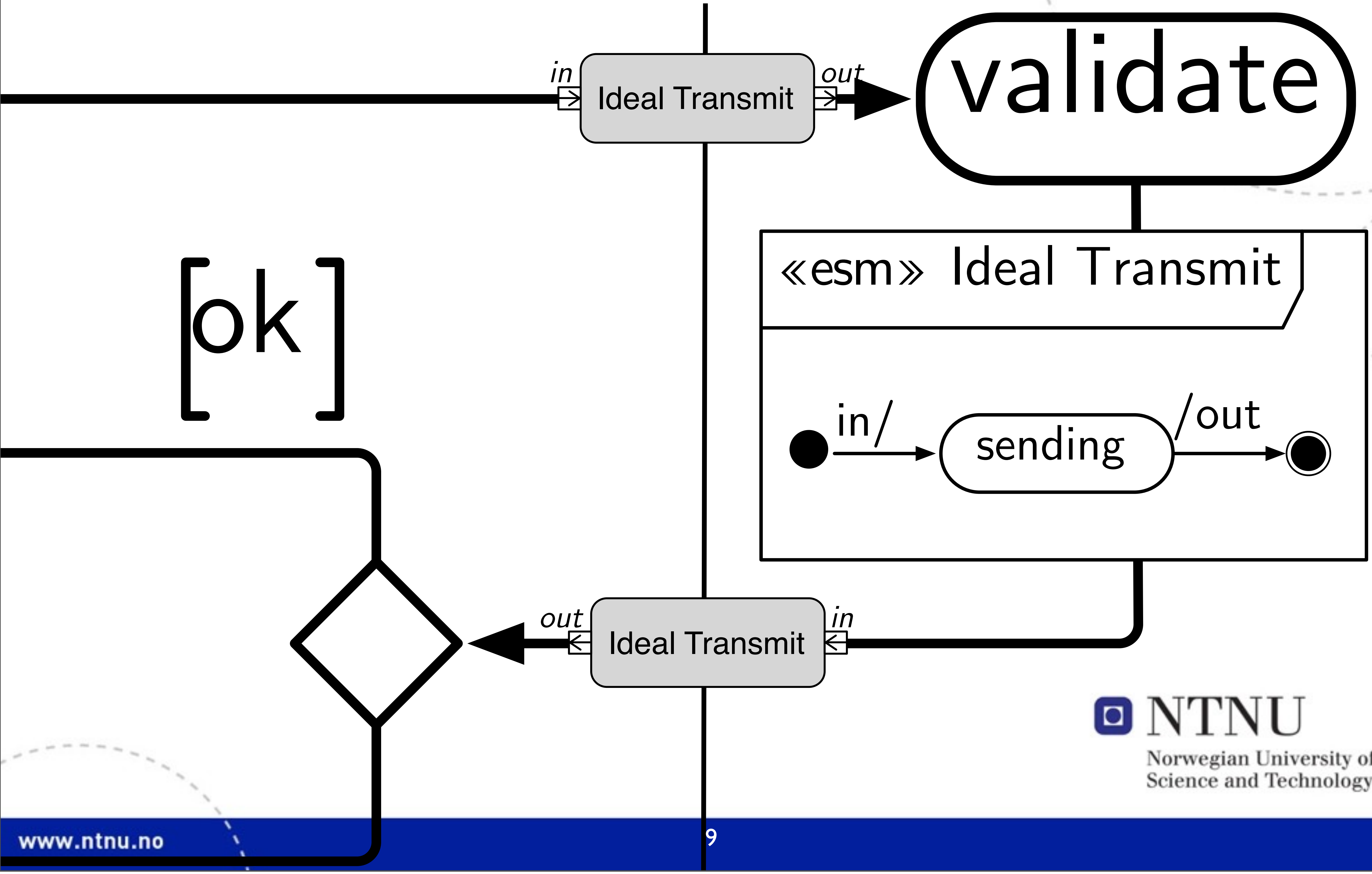
Example – Authenticate



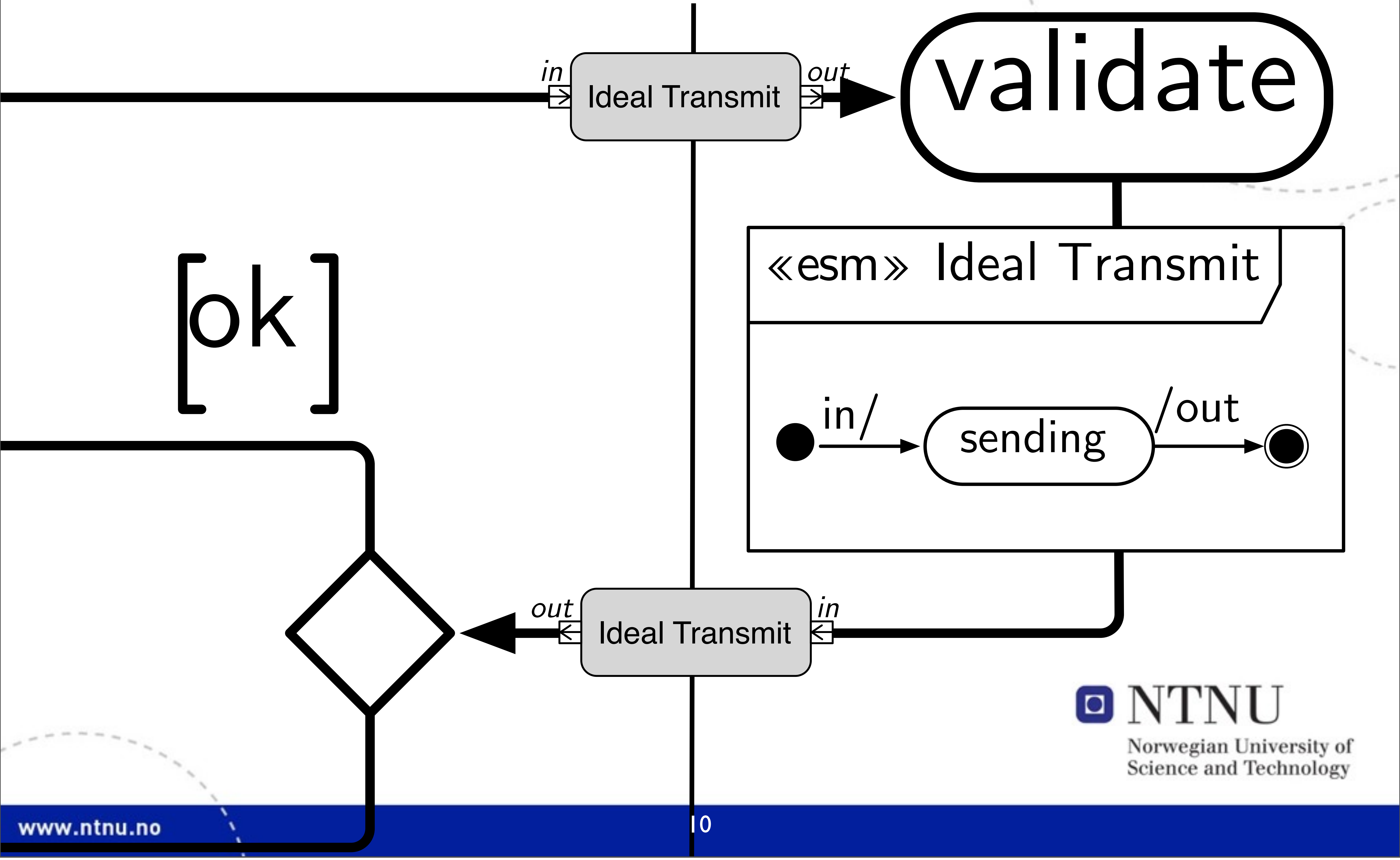


NTNU Norwegian University of Science and Technology

Transmission semantics

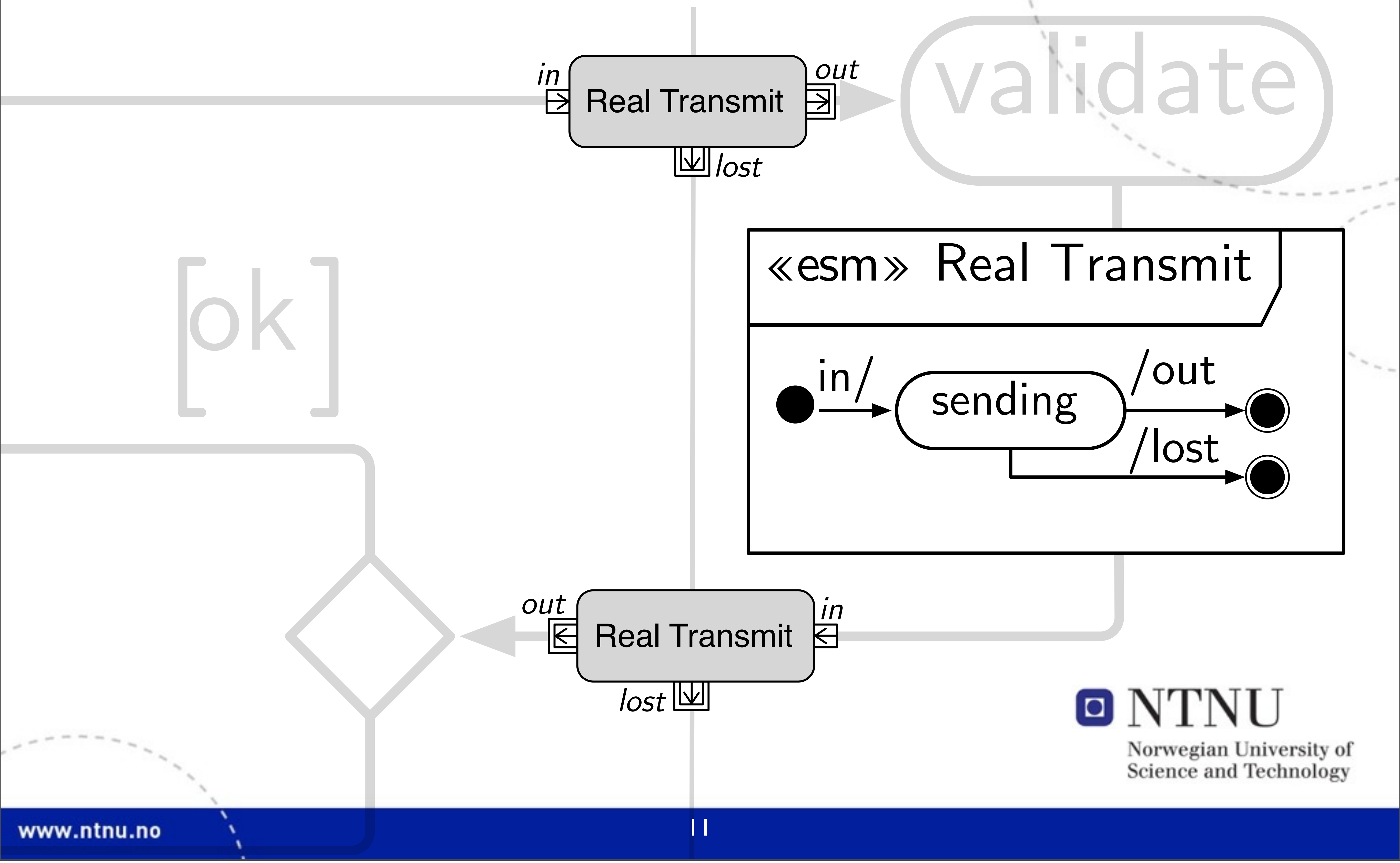


Transmission semantics

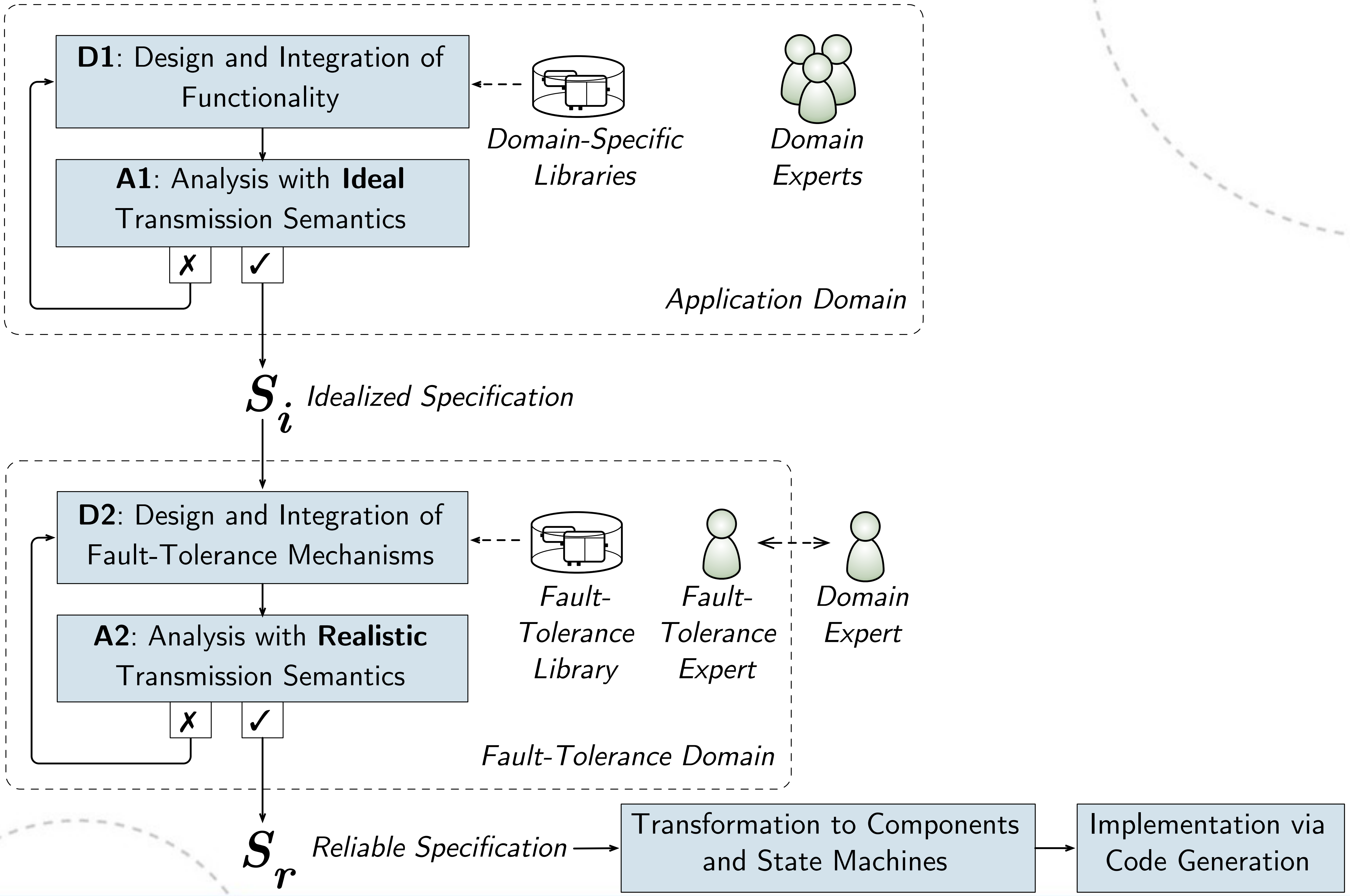


NTNU
Norwegian University of Science and Technology

Transmission semantics

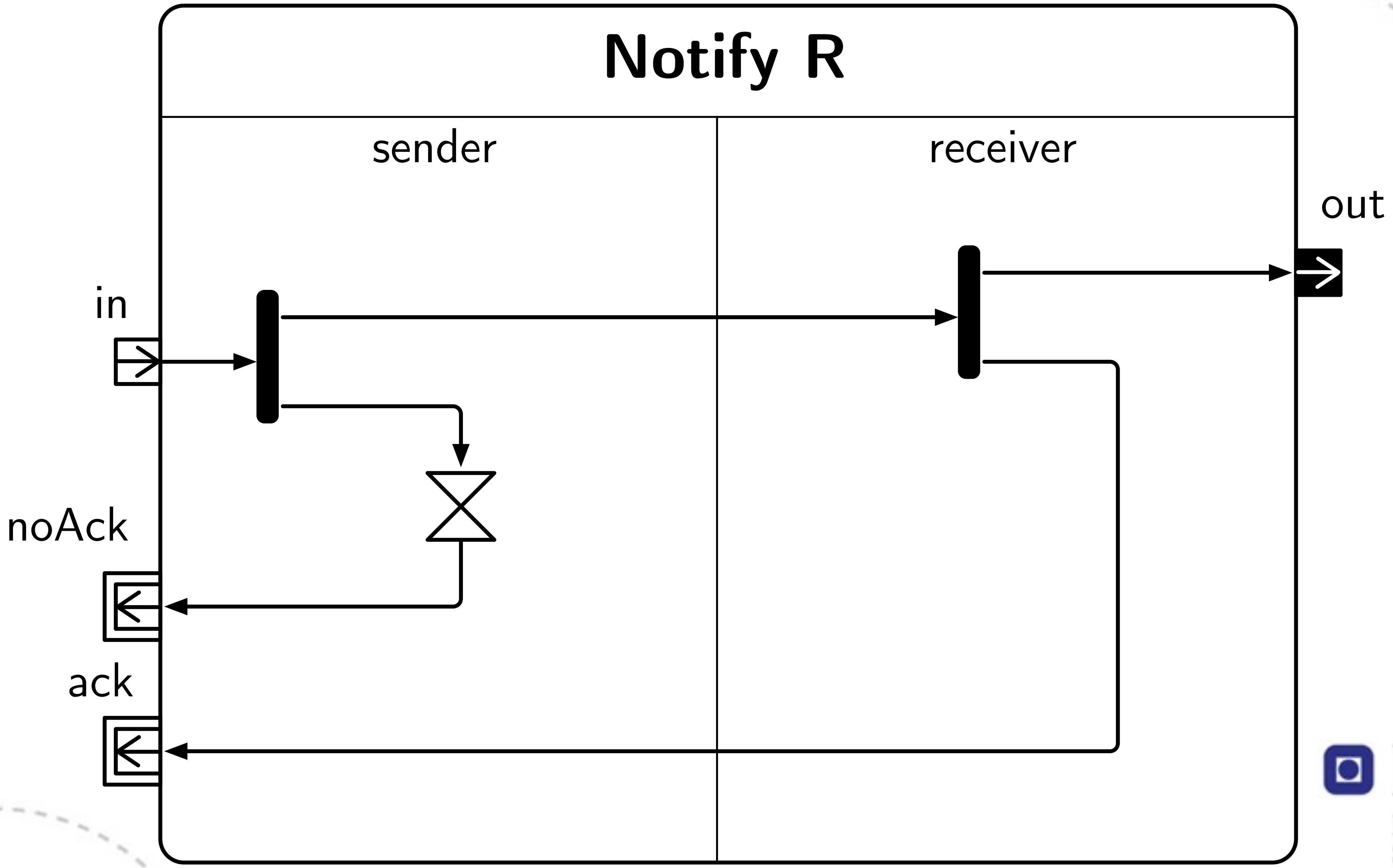


Extended SPACE Method



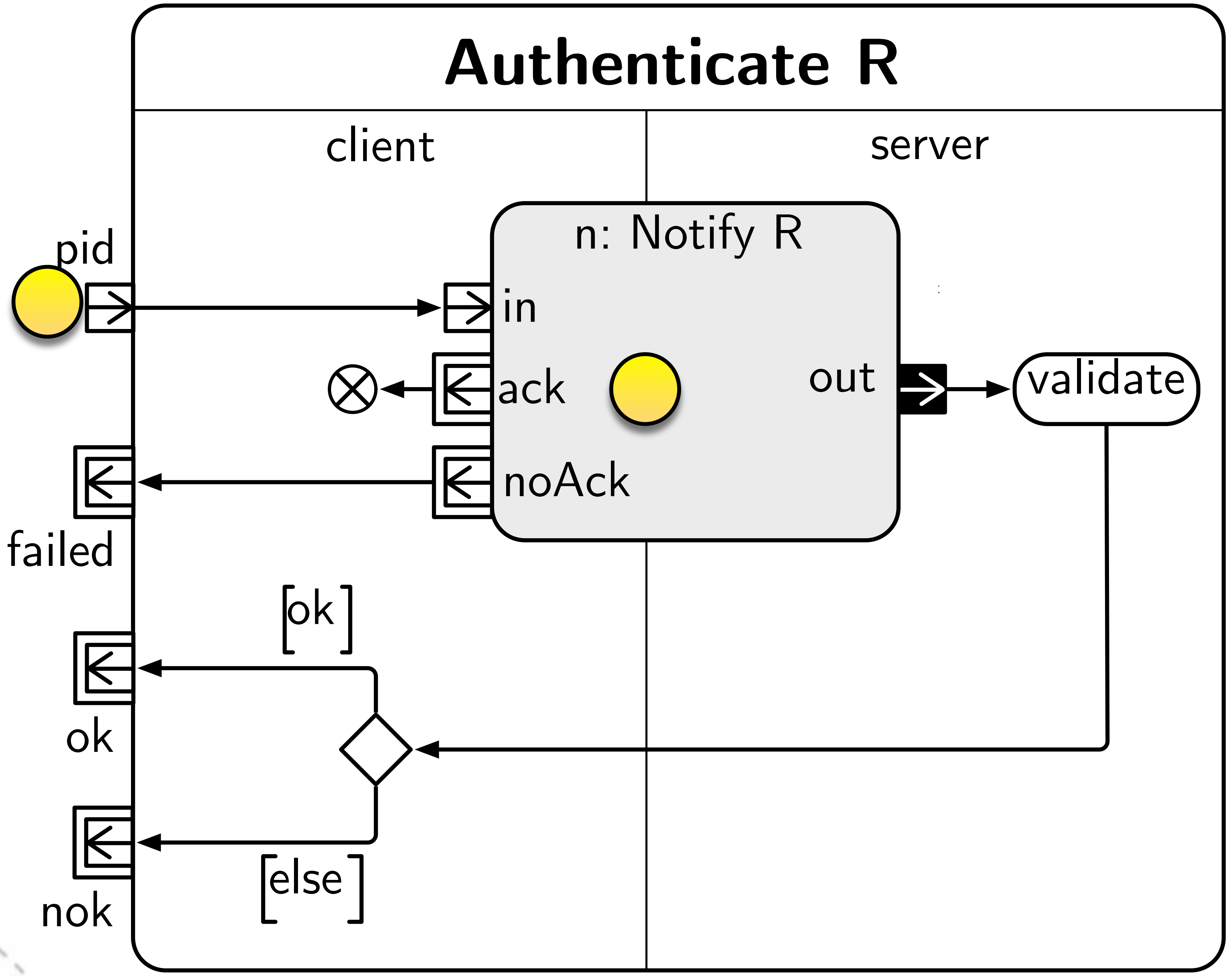
D1: Design and Integration of Functionality
A1: Analysis with **Ideal** Transmission Semantics
✗
✓
Domain-Specific Libraries
Domain Experts
Application Domain
S_i Idealized Specification
D2: Design and Integration of Fault-Tolerance Mechanisms
A2: Analysis with **Realistic** Transmission Semantics
✗
✓
Fault-Tolerance Library
Fault-Tolerance Expert
Domain Expert
Fault-Tolerance Domain
S_r Reliable Specification
Transformation to Components and State Machines
Implementation via Code Generation

Encapsulated message loss detection



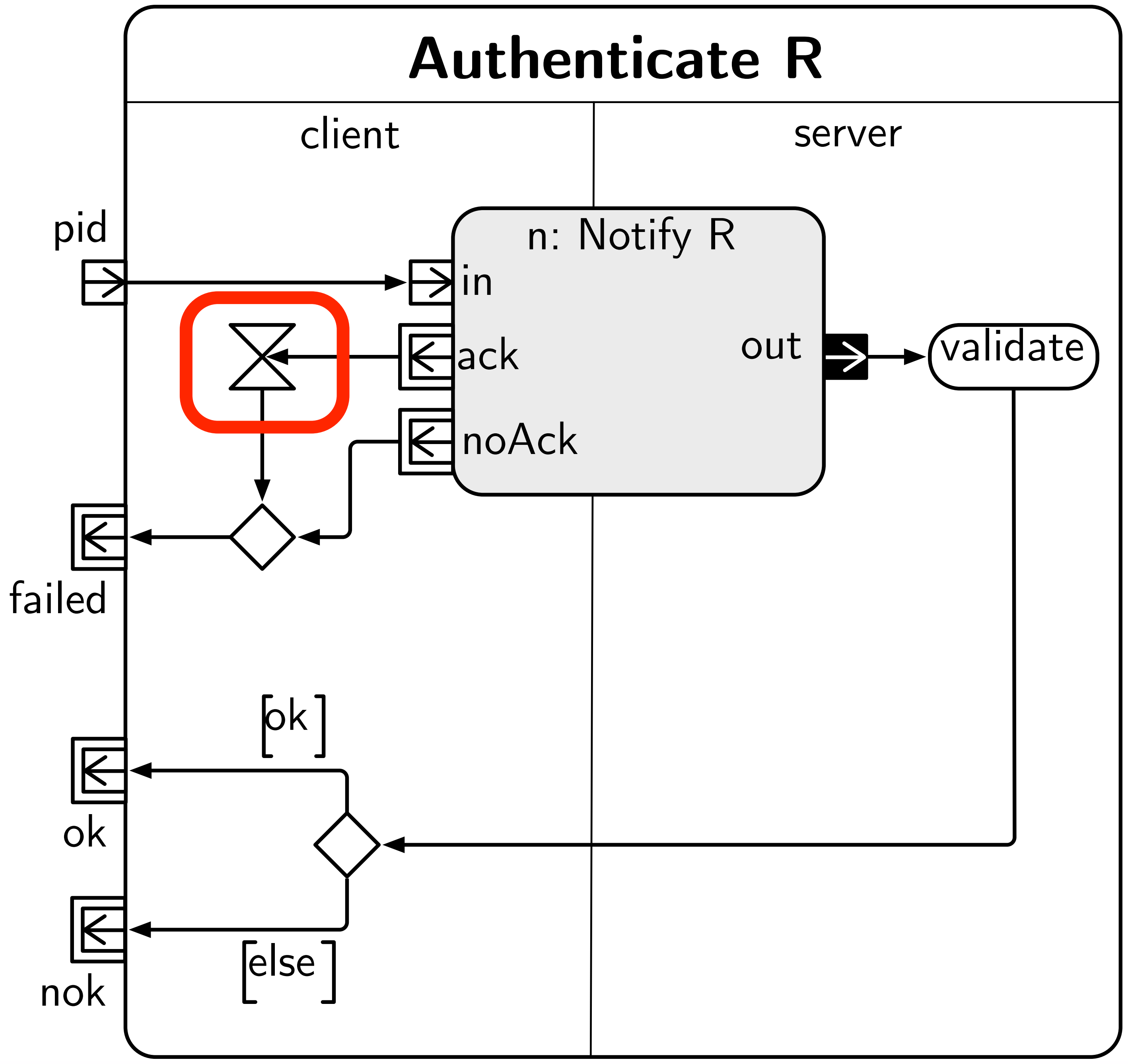
Thursday, 15 April, 2010

Example – Reliable Authenticate (wrong)



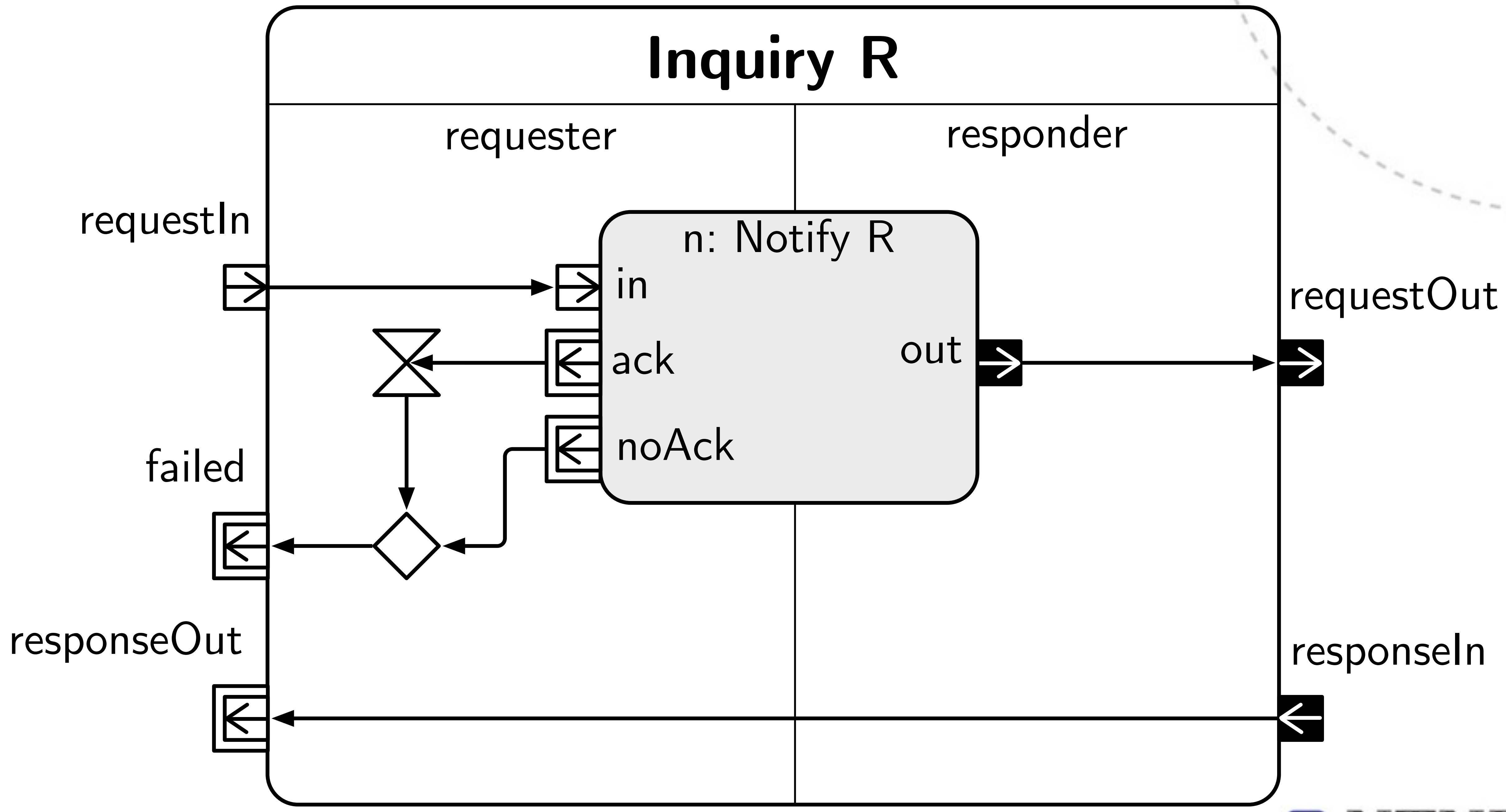
NTNU
Norwegian University of Science and Technology

Example – Reliable Authenticate



NTNU
Norwegian University of Science and Technology

Reliable Inquiry



NTNU
Norwegian University of Science and Technology

Summary

- Problem: Developing reliable, distributed, reactive applications is hard
- Solution:
  - Decompose applications into building blocks encapsulating distribution
  - Allow for an idealized specification (no operational faults) to be developed first
  - Add encapsulated fault-tolerance mechanisms in a second step
  - Use tools for fault removal at every step
- Scope of this paper: Unreliable channels

NTNU
Norwegian University of Science and Technology

Questions for the future

- Should we check application-specific (liveness) properties?
- Is the separation of concerns good enough with bigger systems?
- Can we extend this to software fault tolerance?
- What to put at application layer as building blocks?

NTNU
Norwegian University of Science and Technology