

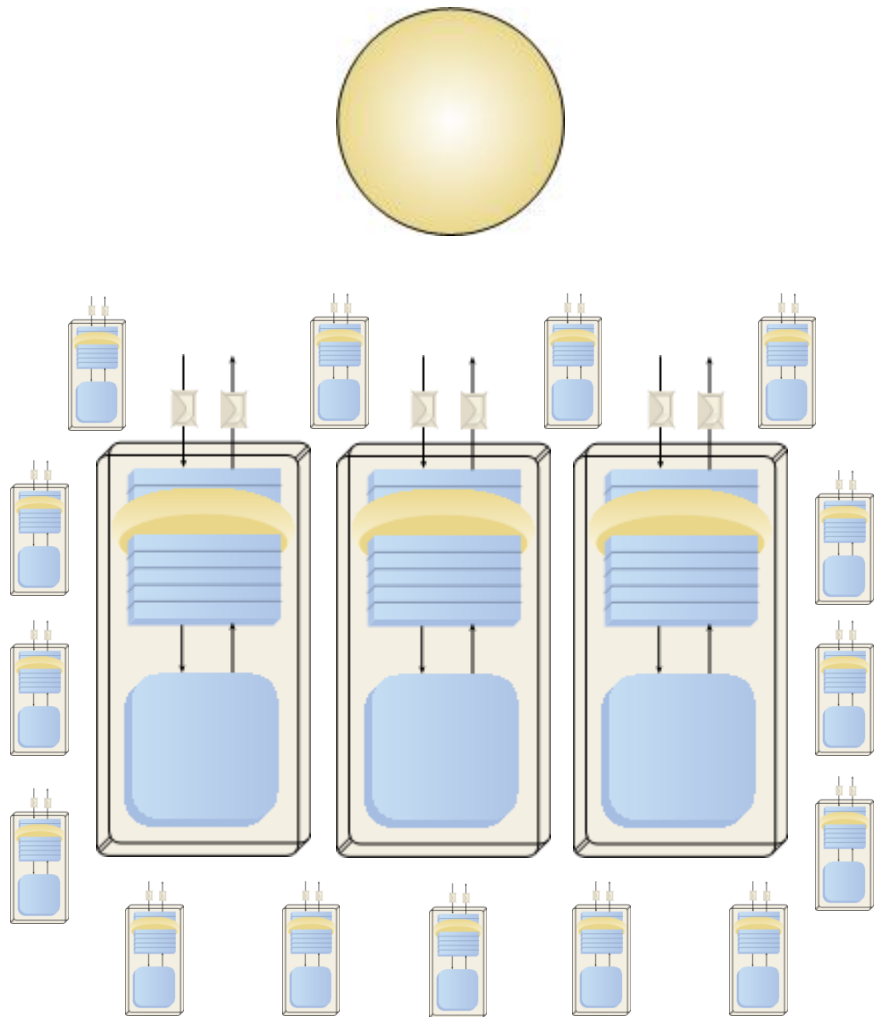
# SOA Design mönster i molnet



Herbjörn Wilhelmsen  
herbjorn.wilhelmsen@objectware.se

OBJECTWARE

# Redundant Implementation



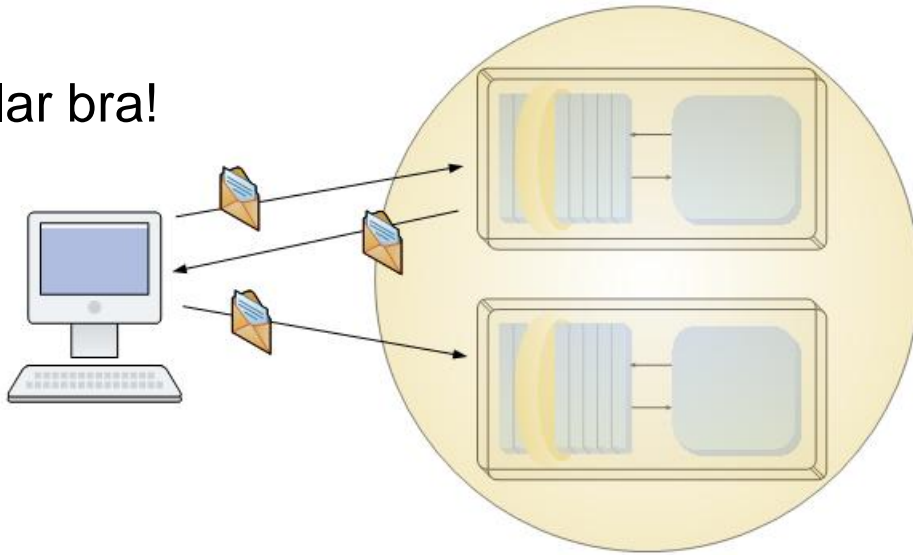
Logisk tjänst

Implementation av tjänst

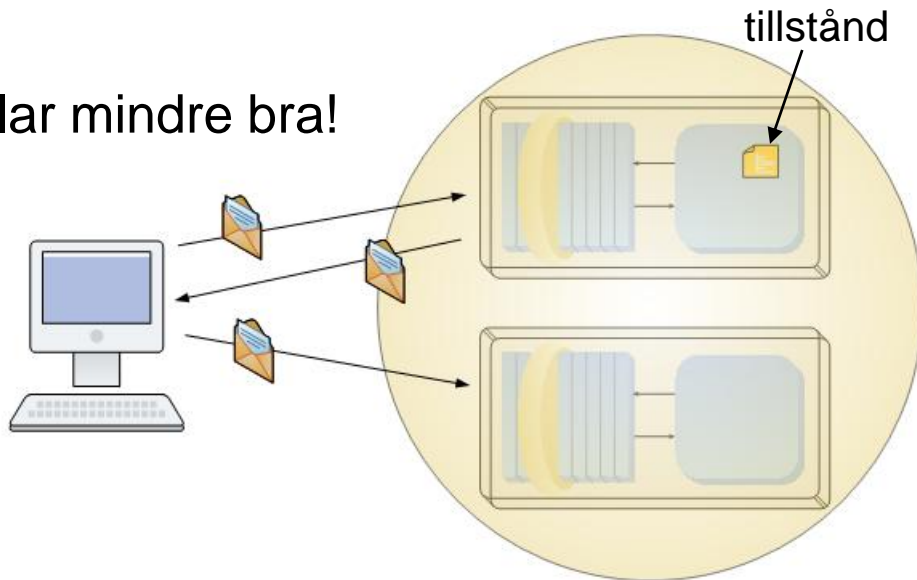
Skala mycket, enkelt och kostnadseffektivt i molnet

# Stateless vs Stateful

Skalar bra!



Skalar mindre bra!



Stateless:

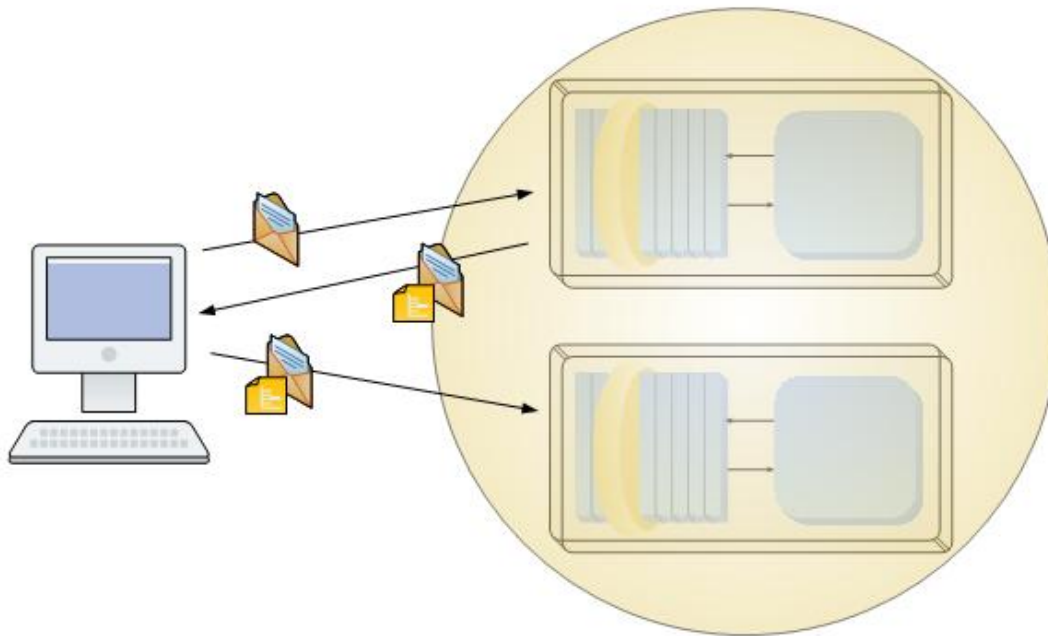
Varje anrop sker utan hänsyn till tidigare anrop

Ibland  
behövs  
tillstånd!

Stateful:

Anrop anrop sker där hänsyn till tidigare anrop måste tas

# State Messaging



- Skicka tillståndet med i meddelandet

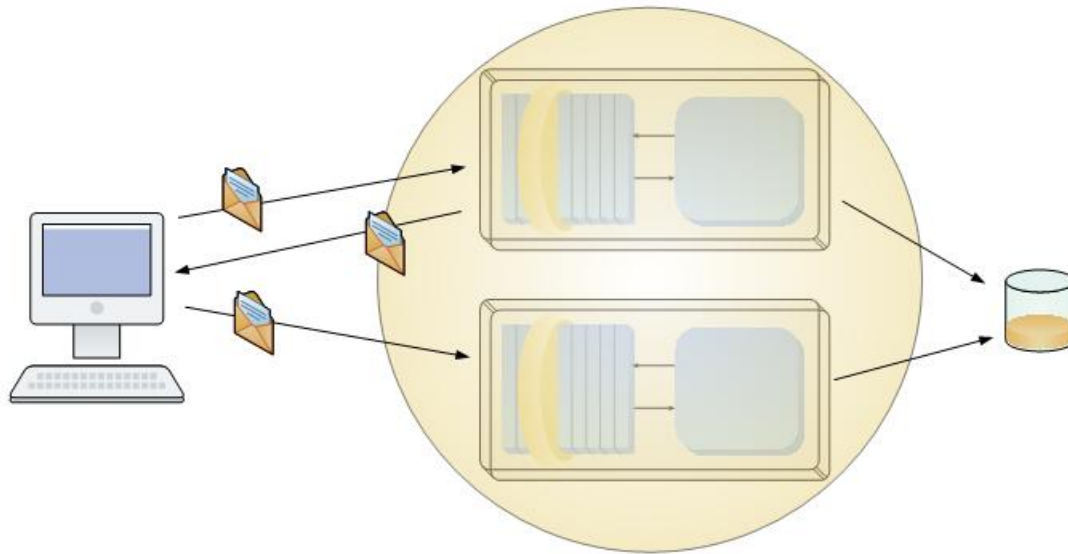
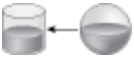
- Skalar lika bra som stateless

- Fungerar inte för
  - delat tillstånd

Fungerar sämre för

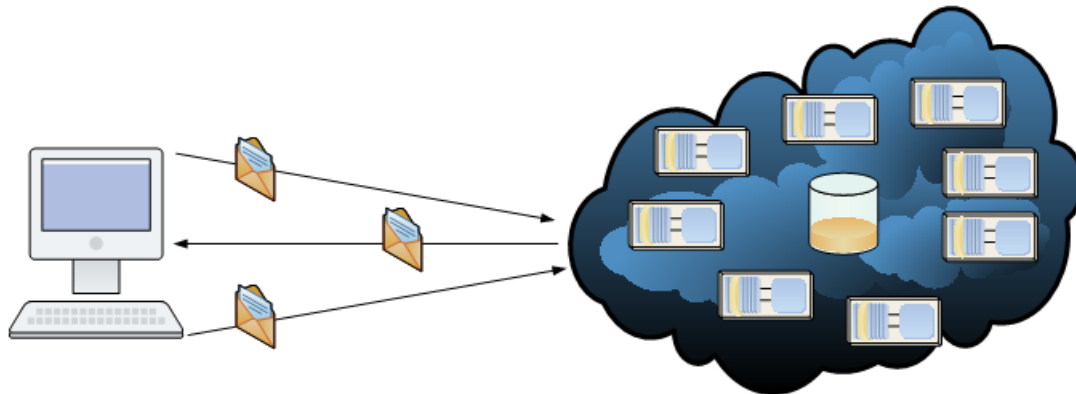
- stora mängder tillstånd
- sekretessbelagt tillstånd

# State Repository (1 av 2)



Fungerar bra för

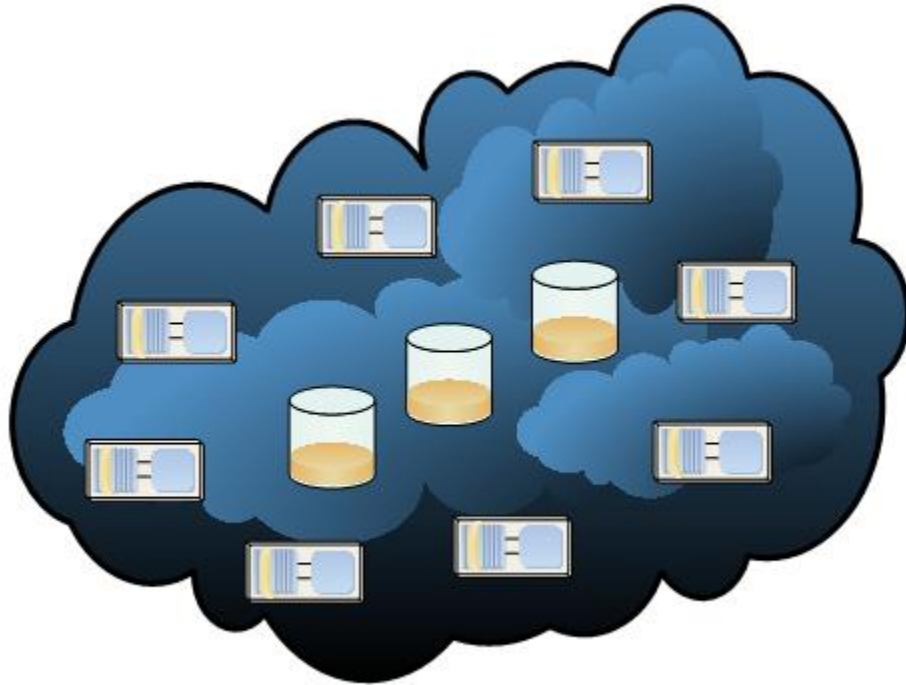
- delat tillstånd
- stora mängder tillstånd
- sekretessbelagt tillstånd



Skalar inte så bra i molnet

- single point-of-failure
- STOR flaskhals

## State Repository (2 av 2)



Flera state repositories:

- ingen single-point-of-failure
- repositories måste synkroniseras

CAP (Consistency-Availability-Partition tolerance teoremet) förstör!

Vi måste välja mellan

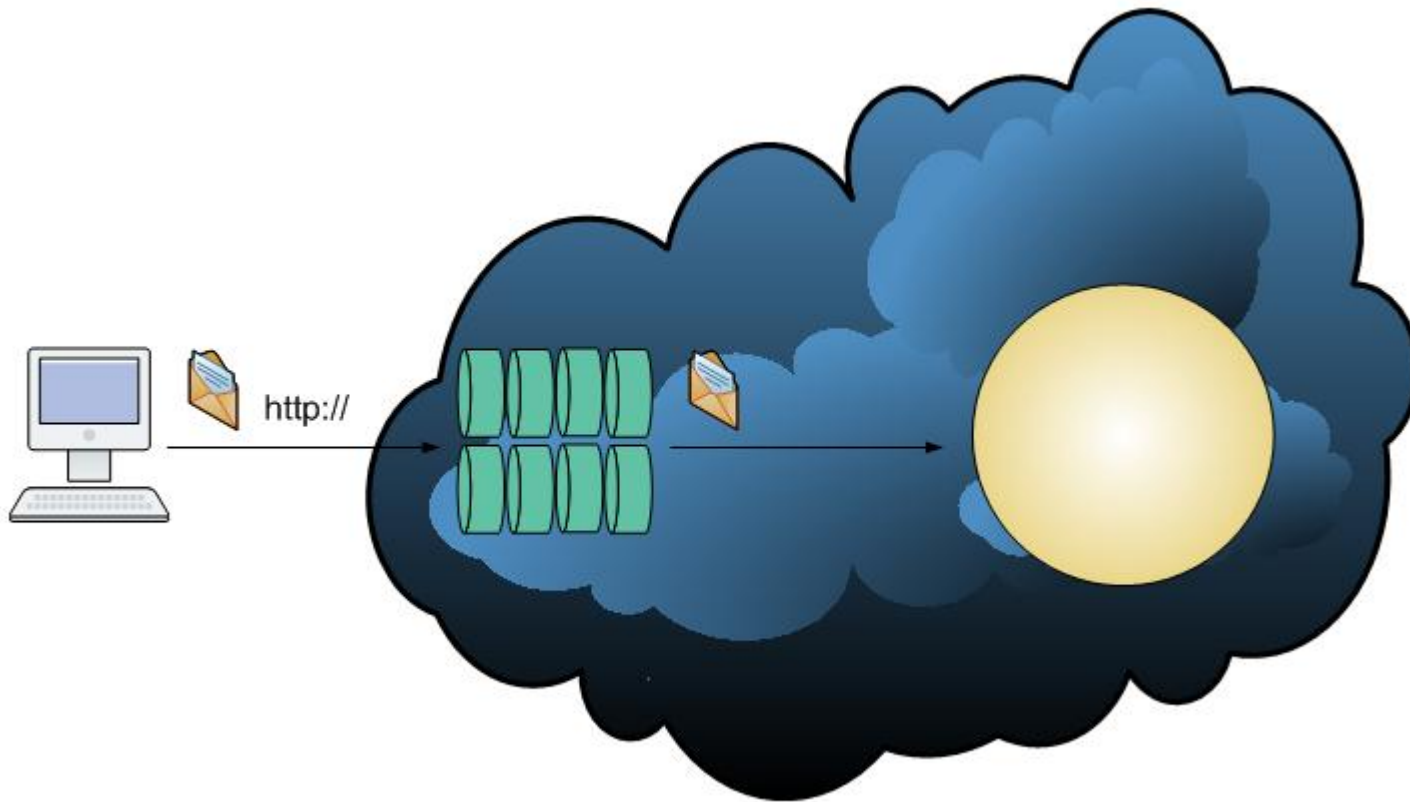
Hög konsistens:

- skriv till 2 av 3
- läs från 2 av 3

Hög tillgänglighet:

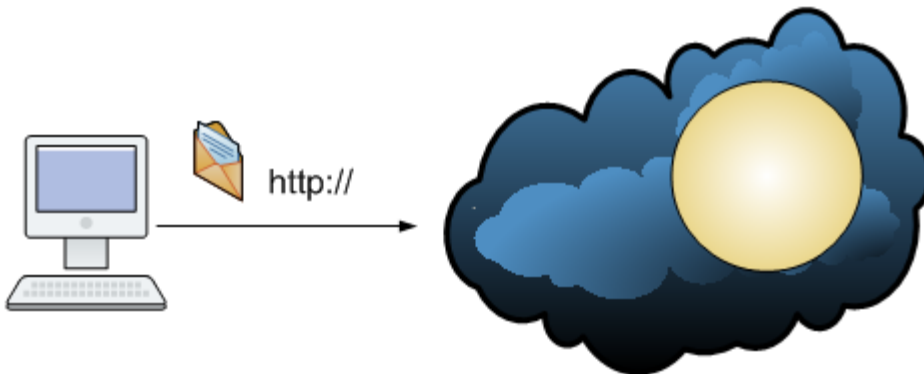
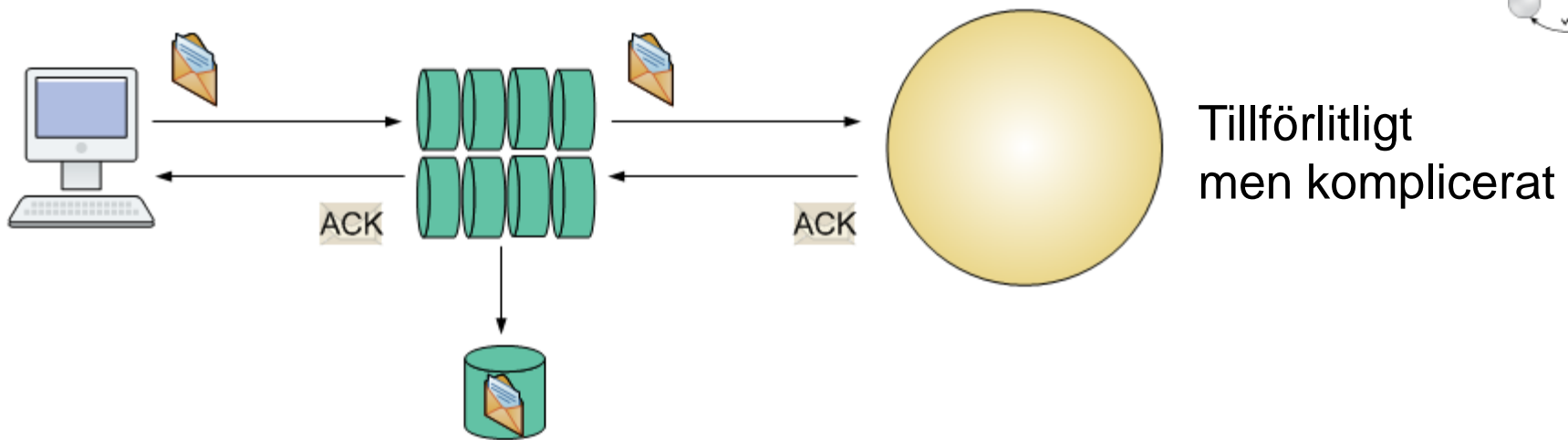
- skriv till 1 av 3
- läst från 1 av 3

# Asynchronous Queuing



- Asynkrona köer kan minska problem med tjänster som är nere
- Köer finns och kan fungera bra i molnet

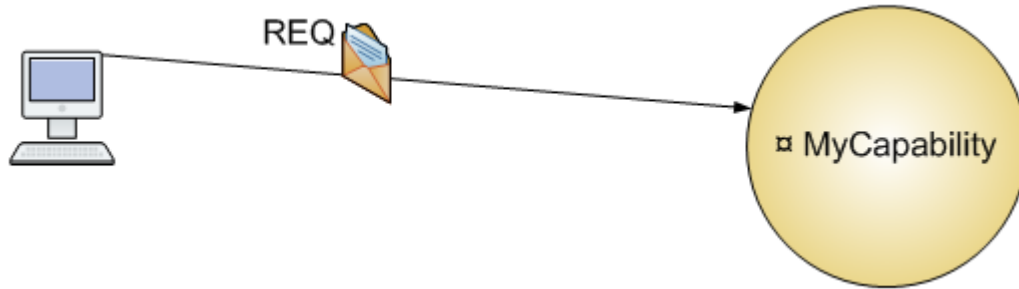
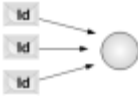
# Reliable Messaging



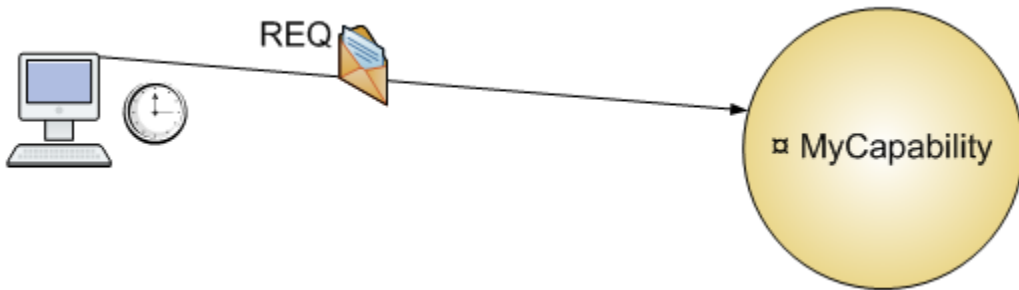
- http är inte tillförlitligt
- svårt att ställa krav på konsumenter när det gäller
  - eget nätverk
  - teknikval



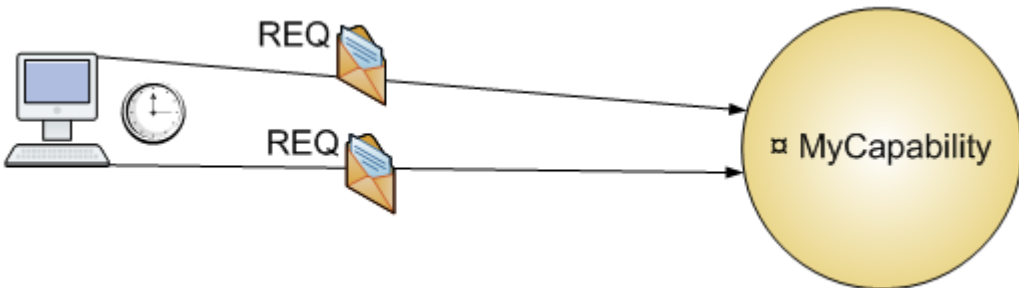
# Idempotent Capability (1 av 3)



Skicka ett  
anropsmeddelande



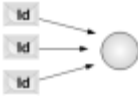
Vänta på ett svar som  
inte kommer...



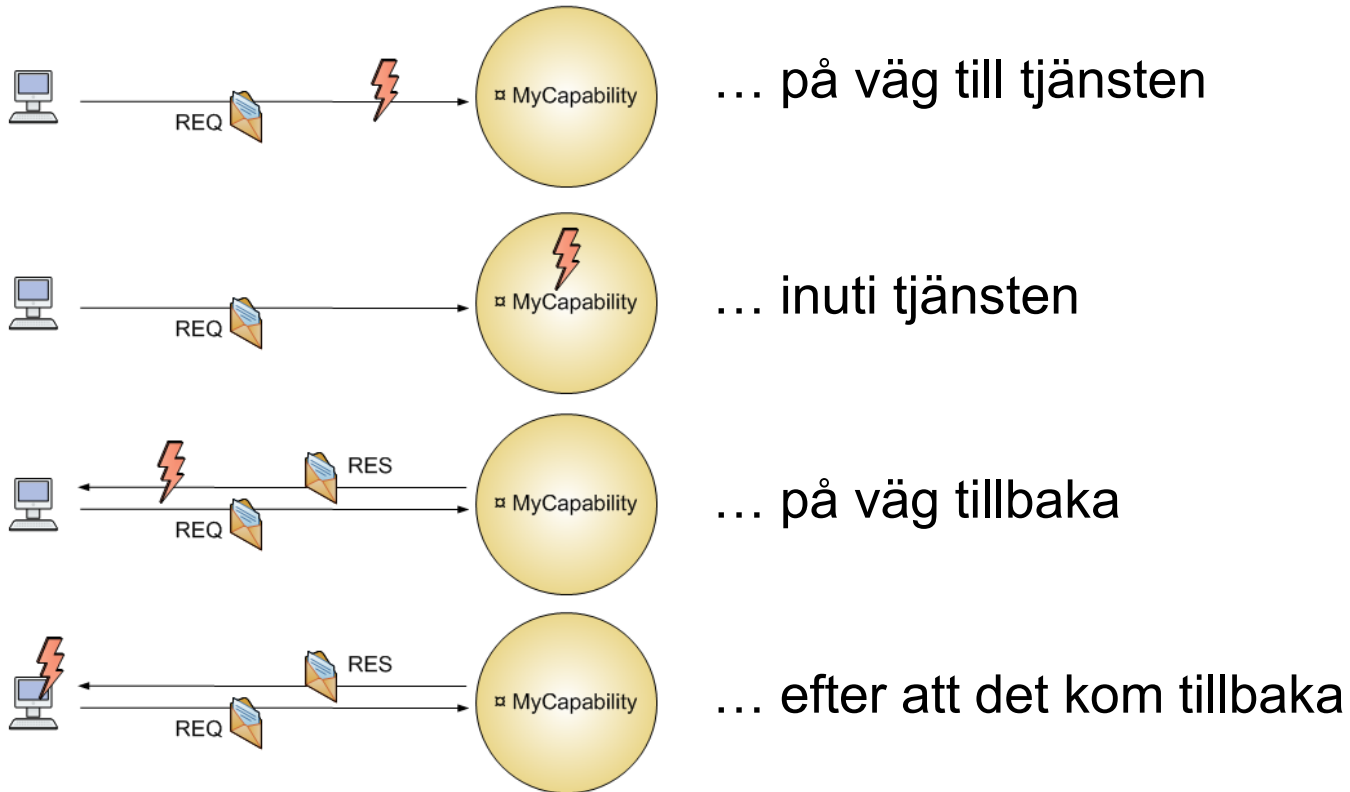
Vad händer sedan?

Att skicka om  
anropsmeddelandet  
skulle kunna vara den  
enda rätta lösningen...

# Idempotent Capability (2 av 3)

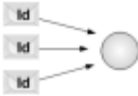


När ett förväntat svar inte kommer tillbaka kan det ha gått fel...



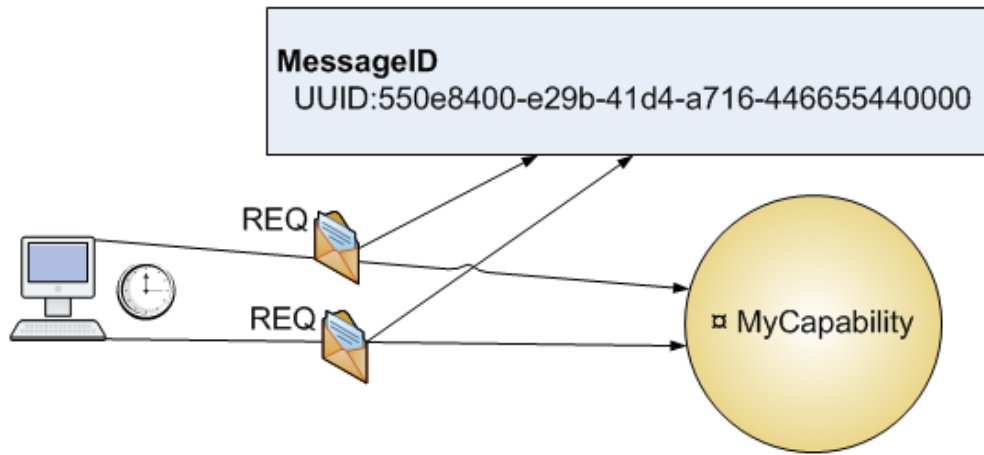
Det spelar stor roll var det gick fel!

# Idempotent Capability (3 av 3)

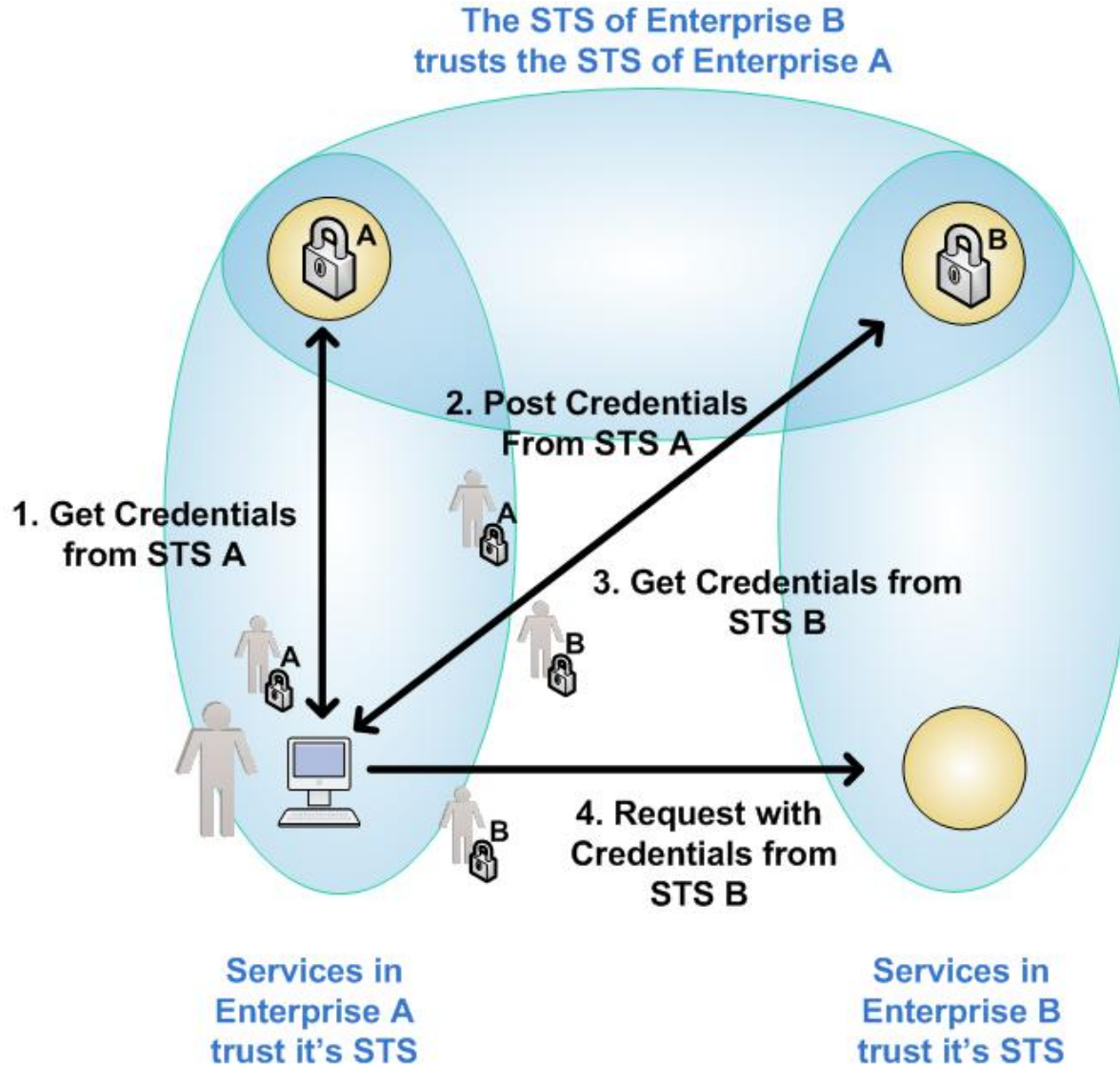


Se till att det inte spelar någon roll hur många gånger ett meddelande skickas:

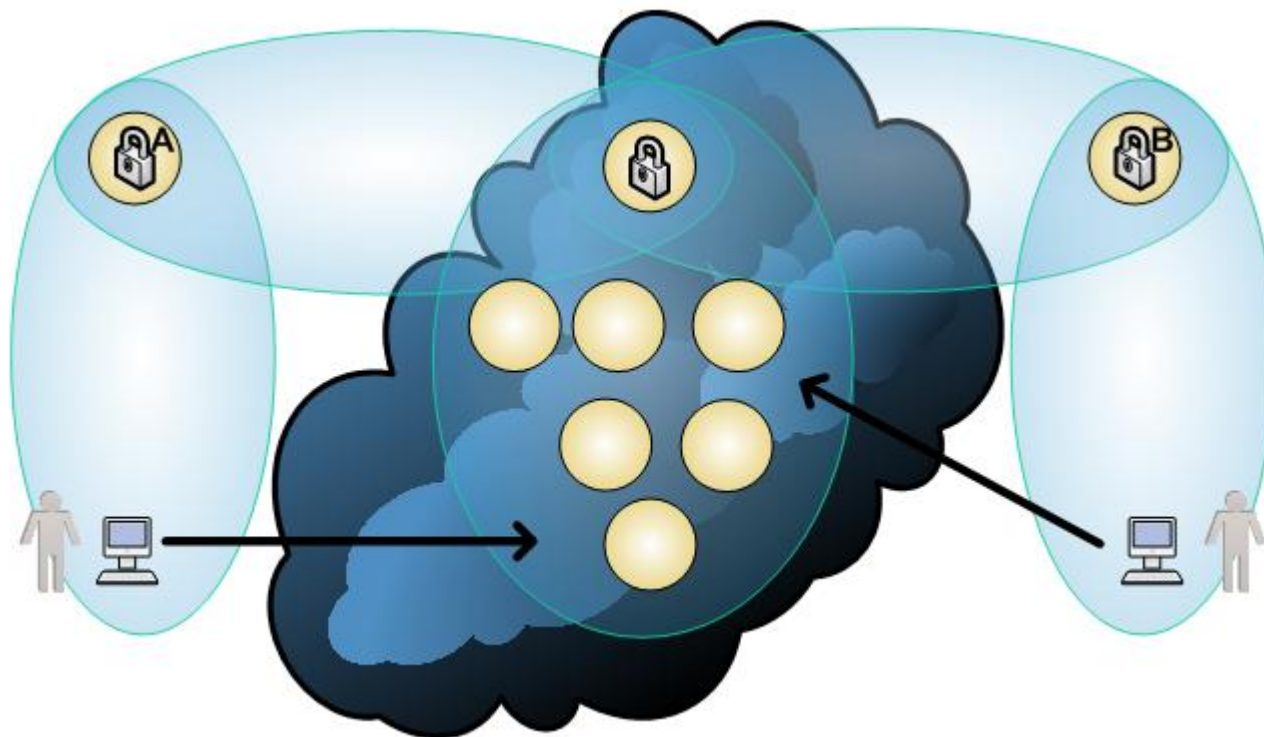
1. Logik som inte har några sidoeffekter alls ← **Kan vara svårt!**
2. Identifiera varje meddelande unikt och spara undan vilka som har behandlats



# Federated Identity (1 av 2)

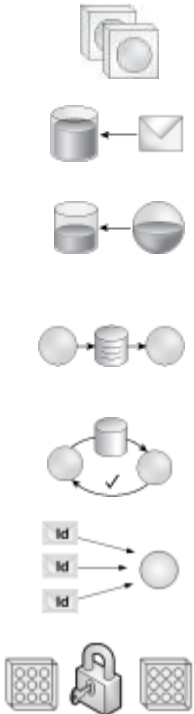


# Federated Identity (2 av 2)



Federated Identity fungerar utmärkt i molnet!

# Frågor?



<http://soapatterns.org>

**SOA Patterns**  
A Community Site for SOA Design Patterns

**Domain Inventory (EI)**

**Overview**

- History
- Achievements
- Podcasts
- Notification Form
- Feedback Form
- Press Release #1
- Press Release #2
- Press Release #3

**Master SOA Design Patterns Catalog**

- Water Pattern List (alphabetical)
- Water Pattern List (by category)
- Water Pattern List with Regu Numbers (PDF)
- Water Pattern List (TUG)
- Pattern Notation
- Pattern Profiles
- Control Legend
- Pattern Contributor Form

**SOA Candidate Patterns**

- Candidate Patterns Overview
- Candidate Patterns List
- Candidate Pattern Contributor Form
- Candidate Pattern Feedback Form

**Design Pattern Basics**

- What's a Design Pattern?
- What's a Design Pattern Language?
- What's a Compound Pattern?

**Supplemental**

- SOA Patterns and Application Technologies
- SOA Design Patterns
- Historical Influence
- SOA Design Patterns and Design Principles
- SOA Design Patterns and Design Granularity

**Resources**

- Design Patterns Publications
- SOA Principles Poster (PDF)
- SOA Principles.com
- wwwSOA.com
- SOA Value Based

**Problem**

How can services be delivered to maximize reposition when enterprise-wide standardization is not possible?

**Solution**

Services can be grouped into manageable, domain-specific service inventories, each of which can be independently standardized, governed, and owned.

**Application**

Inventory domain boundaries need to be carefully established.

**Impacts**

Standardization disparity between domain service inventories imposes transformation requirements and reduces the overall benefit potential of the SOA adoption.

**Principles**

Standardized Service Contract, Service Abstraction, Service Compatibility

**Architecture**

Enterprise, Inventory

**Related Patterns in This Catalog**

Canonical Context (EI), Canonical Resources (EI), Canonical Schema (EI), Contract Centralization (EI), Cross-Domain Utility Layer (EI), Data Model Transformation (EI), Enterprise Inventory (EI), Inventory Endpoint (EI), Logic Centralization (EI), Metadata Centralization (EI), Service Layers (EI), Service Normalization (EI)

**Related Service-Oriented Computing Goals**

Increased Federation, Reduced IT Burden

**Related Publications**

SOA Pattern of the Week: Domain Inventory (9/10/07)

**Quote:** "This pattern is discussed as part of the audio podcast: [Standardizing the Service Inventory and Domain SOA Design Patterns](#)"

**Quote:** "This obligatory element of SOA design patterns will become the standard's least noticed, many organizations will build successful SOA solutions."  
— Steven Brink, Director, Technology Business Unit, Oracle

**SOA Design Patterns**

Thomas Erl  
Foreword by Grady Booch

PRENTICE HALL

With contributions from David Chappell, Jason Hogg, Anish Karmakar, Mark Little, David Orchard, Saladru Roy, Thomas Reichbeck, Arnaud Simon, Clemens Utschig, Dennis Wisnosky, and others

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL

SOA Books | SOA Resources | What's SOA? | SOA Principles | SOA Standards | SOA Glossary

Copyright © 2007-2009, SOA Systems, Inc.

