# ISU: 5. Message Distribution System

## Messaging distribution system - Why & how?

* For at skabe større afkobling
  + Altså mulighed for at kommunikere uden på forhånd at ”kende hinanden” eller have en delt message queue
  + Også for at skabe afkobling i flow – så kommunikerende dele ikke står og venter på et svar eller venter på at en modtager, modtager beskeden
  + Afkobling i tid – så kommunikerende dele ikke behøver at være online på samme tid eller i en bestemt rækkefølge
* Før MD har vi med MQ været nødt til at passe pointere til MQs rundt ved programstart
  + Kyllingen og ægget

Det kan laves på flere måder:

* Postoffice – specifik modtager
* Brodcasting – modtager ukendt

## The PostOffice design - Why and how?

Posthus metoden er ligesom et posthus:

* En besked sendes til posthuset, med information om hvem der skal modtage beskeden – specifik modtager – men de kommunikerende dele er altså ikke nødt til at kende hinanden!
* Posthuset modtager beskeden og sender den til rette modtager
* Modtageren modtager beskeden via en message queue

## Decoupling achieved

* Afkoblet i space
* Time – not so much
* Flow: producer er asynkront, consumer er blokerende(synkront)

## Design considerations & implementation

Krav:

* Der skal laves et posthus
* Dette nødvendiggør at modtager tråden har givet besked til posthuset om hvordan beskeder til modtager tråden, skal sendes til modtager tråden 🡪 altså passing af message queue

Problemstilling:

* **Hvad vil der ske hvis en besked sendes til en modtager tråd, der endnu ikke har fortalt posthuset ”hvor den bor”?**
* **Og hvad sker der hvis posthuset ikke er initialiseret?**
* Global identifier på besked
* One way communication

## Singleton Pattern

**Broadcasting:**

* Her er det ikke nødvendigt at vide hvem modtageren er
* En publisher broadcaster information til et MDS
* Og consumers subscriber på den information de vil modtage fra MDS
  + Subscriber med message queue og ID beskeden skal modtages med
* En fordel er at flere kan subscribe på den samme information
* I forbindelse med message distribution system skal der passes mange pointere/referencer for at alle kan få adgang til MDS
  + Kan klares med singleton

**Singleton:**

* Simpelt kode – static block initialization
* ***The singleton pattern is implemented by creating a*** [***class***](http://en.wikipedia.org/wiki/Class_(computer_science)) ***with a method that creates a new instance of the class if one does not exist. If an instance already exists, it simply returns a reference to that object. To make sure that the object cannot be instantiated any other way, the*** [***constructor***](http://en.wikipedia.org/wiki/Constructor_(computer_science)) ***is made*** [***private***](http://en.wikipedia.org/wiki/Private_member)
* Kun ÉN instantiering af klassen
* **Tilgængeligt for alle**
  + Der alle bruger den samme instans
* ”First acces creates”
  + Både godt og skidt – multitrådsproblem – hvem kommer først?
  + ”Double-checked locking idiom” virker ikke????
  + *The singleton pattern must be carefully constructed in* [*multi-threaded*](http://en.wikipedia.org/wiki/Thread_(computer_science)) *applications. If two threads are to execute the creation method at the same time when a singleton does not yet exist, they both must check for an instance of the singleton and then only one should create the new one. If the programming language has concurrent processing capabilities the method should be constructed to execute as a mutually exclusive operation.*
* Anden fordel: Derudover kan det laves så der ikke skal bruges låse

Afkobling:

* Hvis det er lavet korrekt kan der afkobles i alle 3 dimensioner

Ulempe: Globalt – ikke tøjlet – du kan ikke spore hvem der egentligt bruger den

## Publisher/Subscriber schemes - which one?

* Topic based
  + Identifikation via et keyword (string)
  + Subscriber og publisher til et givent keyword
    - Wildcards 🡪 match a given set of keywords
* Content based
  + Identifikation via properties af en given notifikation (string)
  + Subscriber til noget bestemt indhold
* Type based
  + Identifikation via type
  + Type sikkerhed