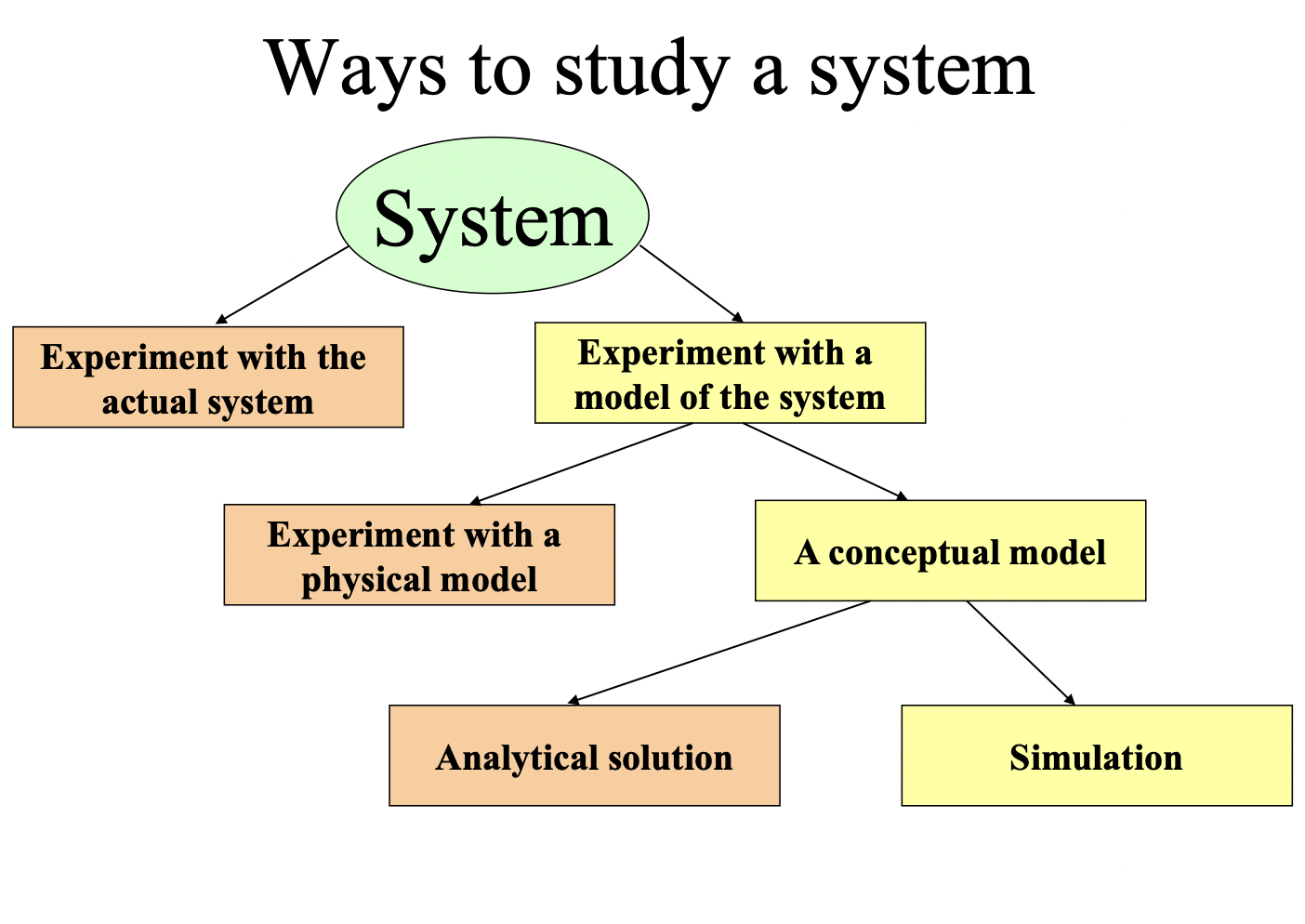
*Föreläsning 7*

**Simulering**

* Imitering av ett system
* Vi analyserar systemet och får ut ett antal parametrar



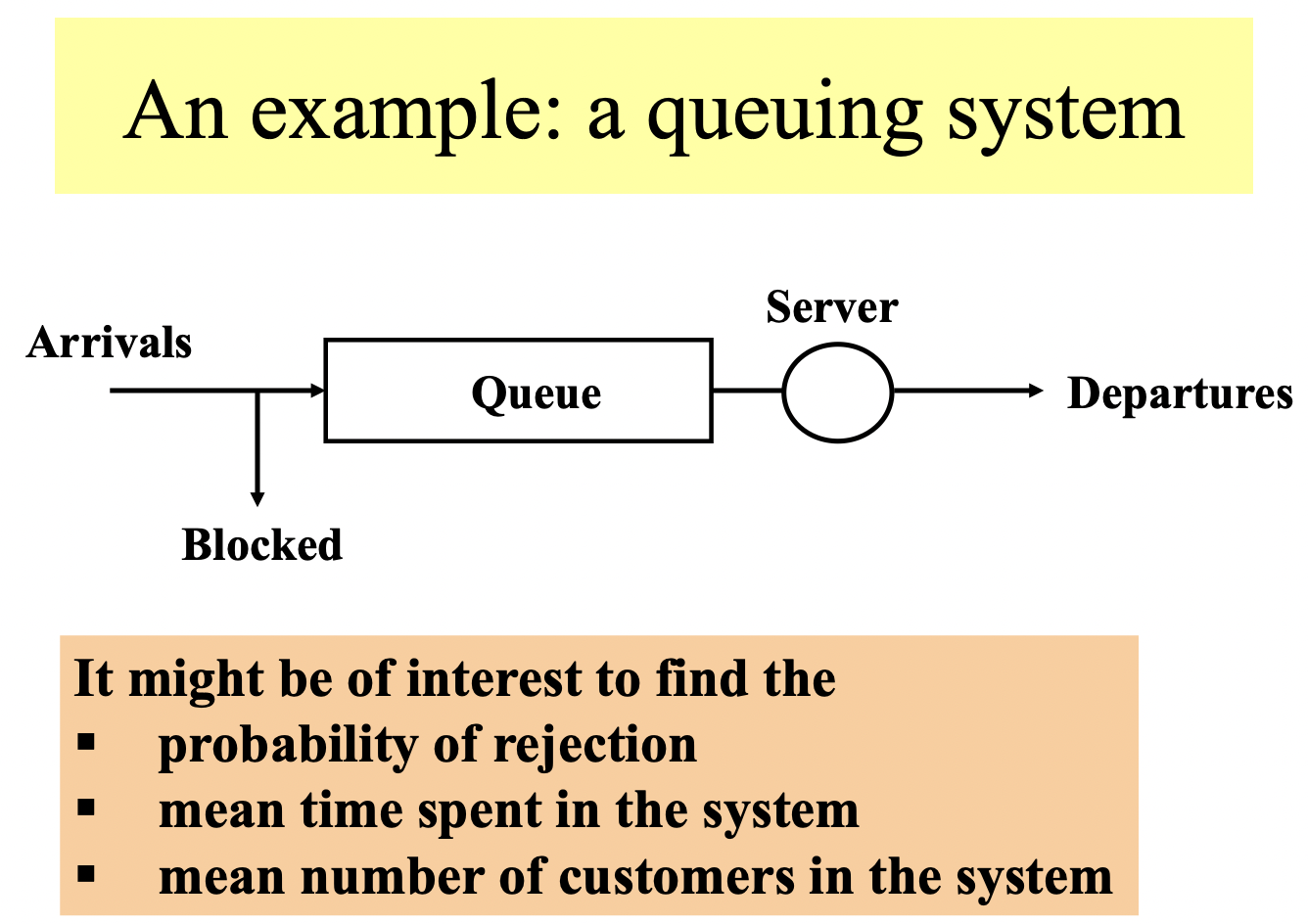
* Olika system
  + Kontinuerliga system - Något som kan anta vilket värde som helst
    - Görs oftast med diffekvationer
  + Diskreta system
    - Köer
  + Hybrida system
    - När vi analyserar ett kösystem vill vi se hur många som finns i kön (diskret) och hur lång tid varje person är i kön (kontinuerligt)
* Fördelar
  + Förutspå vad som händer vid förändring
  + Analysera en liten detalj vid förändring
  + Finna flaskhalsar vid exempelvis köbildning
* Nackdelar
  + Avancerat
  + Tidskrävande
  + Begränsningar i noggranhet
* Modelling concepts (discrete system)
  + Tillståndsvariabler
  + Händelser
  + Regler - Vad gör vi när det kommer en ny kund?

**Two approaches to simulation**

**1. Händelsesimulering**

* Här hanterar vi **ENDAST** **1** **händelse**
* “Här kommer en ankomst, nu ska vi allt för att få in detta i systemet”
* **Följande behövs/krävs (metod)**
  + **1. Beskrivning (av ett tillstånd)**
  + **2. Händelser**
  + **3. Regler**
* Vi lägger händelser i en lista
  + Har en viss tid och ett antal parametrar (ex om paket prioriteras)
  + Vi tar alla dessa händelse och sorterar på de tider när de ska inträffa
* Hur gör vi?
  + 1. Plocka ut första elementet i listan
  + 2. Sätt tiden
  + 3. Uppdatera systemet
  + 4. Om EJ avslutat, gå vidare till steg 1!

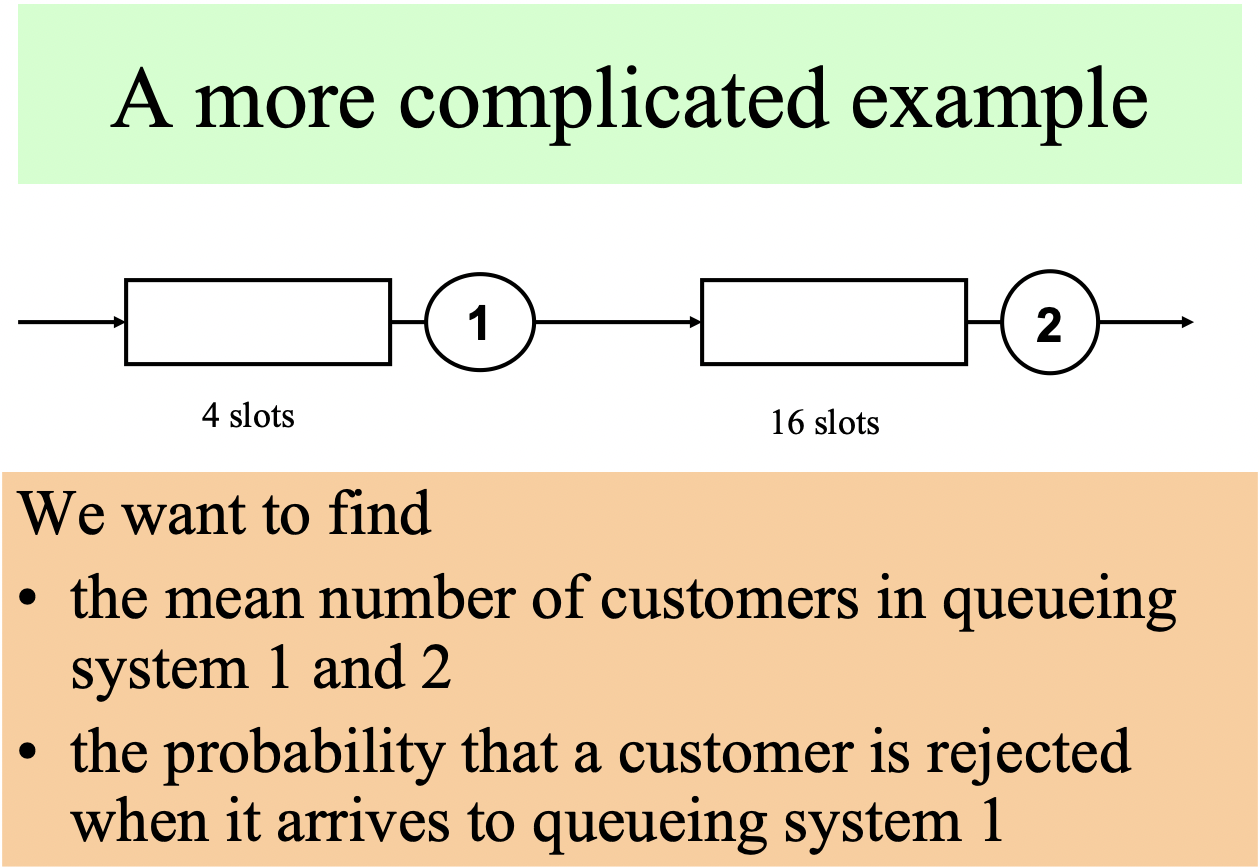
**1.1 Ett exempel**



* **Vad vill vi veta?**
  + Medelantal kunder i system 1

|  | **3 SAKER BEHÖVS** |  |
| --- | --- | --- |
|  | **1. TILLSTÅNDSBESKRIVNING**   * N, antal kunder i systemet |  |
|  | **2. HÄNDELSER**   * Ankomst * Avgång * Mätning (ändrar EJ systemet/tillstånd) * OBS! Mätningar görs slumpmässigt, dvs ALDRIG mätning med exakta tidsavstånd mellan varje mätning!!! |  |
|  | **3. REGLER**   * Medeltid mellan ankomster * Betjäningstid exponentialfördelad * Antal kunder i kön? |  |
|  | **Pseudokod till ovanstående problem**    a, lambda  s, my |  |
|  |  |  |

**1.1 Ett svårare exempel**



* **Vad vill vi veta?**
  + Medelantal kunder i system 1 och 2
  + Vad är sannolikheten att en kund som kommer i system 1/2 blir spärrad?
    - Kan skapas kö mellan de två betjänarna, givet att betjänare 2 är slöare än 1

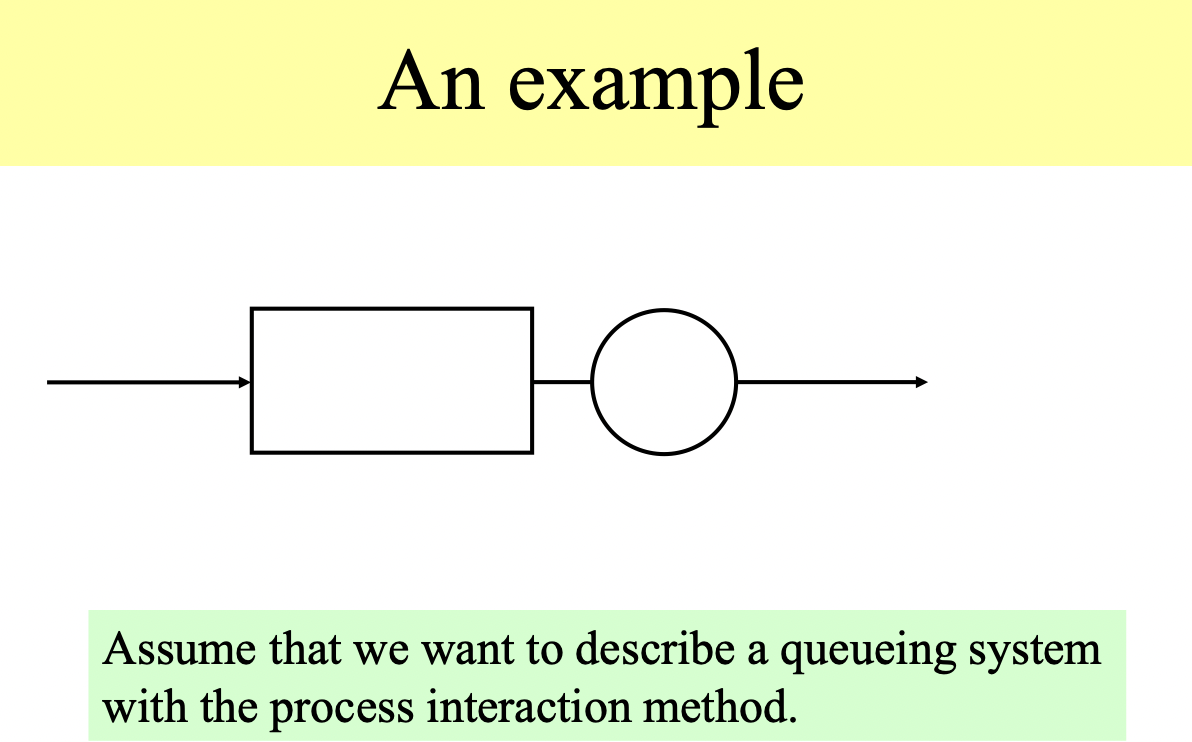
|  | **3 SAKER BEHÖVS** |  |
| --- | --- | --- |
|  | **1. TILLSTÅNDSBESKRIVNING**   * Tillståndsvariabler (=finns inne i systemet)   + N1 = antal kunder i kösystem 1   + N2 = antal kunder i kösystem 2 * Mätvariabler (Inga tillståndsvariabler)   + Antal ankomster   + Antal spärrade |  |
|  | **2. HÄNDELSER**   * Ankomst av kund i system 1 * Avgång av en kund i system 1 **&** Ankomst av en kund i system 2 = SAMMA händelse! * Avgång av en kund i system 2 * Mätning |  |
|  | **3. REGLER** (3 stycken)  **Regel för ankomst till 1**  void RuleArrivalTo1(){  NoOfArrivals++;  If (N1 < 4)  N1++;  else  NoRejected++;  If (N1 == 1)  InsertEvent(DepartureFrom1, time + 0.2);  InsertEvent(ArrivalTo1, time + nextArrival());  }  **Regel för avgång från 1 = Ankomst i 2 (OBS!)**  void RuleDepartureFrom1{  N1--;  if (N2 < 16)  N2++;  if (N2 == 1)  InsertEvent(DepartureFrom2, time + 0.1);  if (N1 > 0) then  InsertEvent(DepartureFrom1, time + 0.2);  }  **Regel för avgång från 2**  void RuleDepartureFrom2{  N2--;  if (N2 > 0)  InsertEvent(DepartureFrom2, time + 0.1);  }  **Regel för mätning**  void RuleMeasurement{  write(file1, N1); //SKRIV IN ANTAL I SYSTEM 1  write(file2, N2); //SKRIV IN ANTAL I SYSTEM 2  InsertEvent(Measurement, time +  NextMeasurement());  } |  |
|  |  |  |

YTTERLIGARE ETT EXEMPEL

* Sannolikheten att en kund behöver mer än 5 sekunder i systemet??
* Vi måste ha reda på när kunderna kommer och när de går
  + Lagra data i en lista
  + Kan implementeras med en double linked list och en vektor
* Kommer INTE göras i kursen!

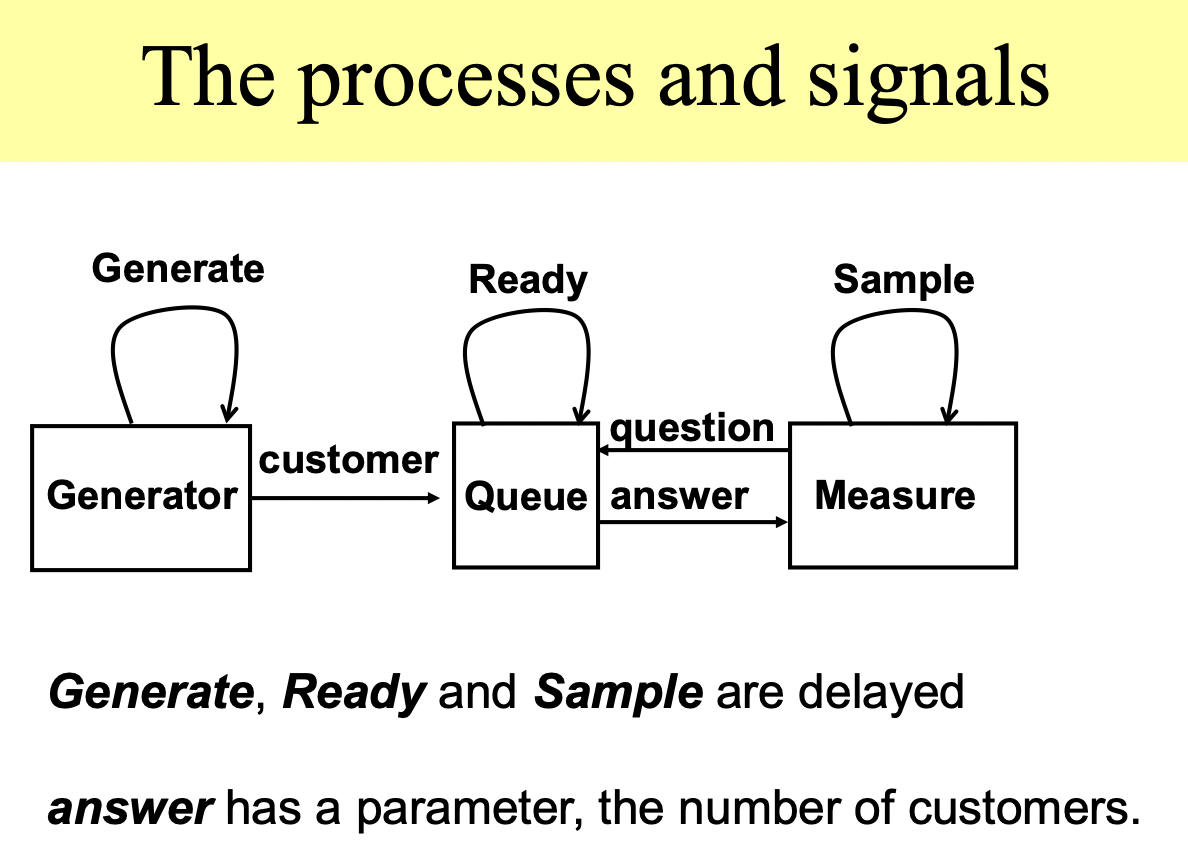
**2. Processimulering**

* En process som sköter en del
  + Radiolänk
  + Fyllning av routingtabell
  + Har ett intern tillstånd
  + Kommunicerar med andra genom utskick av signaler
    - Signaler har namn och kan ha data i sig
  + Under aktiviteten kan ex parametrar uppdateras eller andra signaler kan skickas



**Vilka processer behövs? (“Glöm händelse-tänket”)**

* Process som hanterar kösystemet (**Queue**)
* Process som generar nya kunder (**Generator**)
* Process som mäter (**Measure**)



|  |
| --- |
| **Process för Generator**  if received signal = generate {  //Ny signal som skickar iväg ny kund till queue  SendSignal(customer, Queue, time);  //Ny signal till sig “själv”  SendSignal(generate, Generator, time + Exp(4));  } |
| **Process för Queue**  if received signal = customer{  N := N + 1;  if N = 1 then  SendSignal(ready, Queue, time + Exp(2));  }  //Signal som skickats till oss själv  else if received signal = ready{  N := N – 1;  if N > 0 then  SendSignal(ready, Queue, time + Exp(2));  }  //Förfrågan från mätprocessen (Measure)  else if received signal = question{  SendSignal(answer(N), Measure, time);  } |
| **Process för Measure**  If received signal = sample {  SendSignal(question, Queue, time);  SendSignal(sample, Measure, time + Exp(10));  }  //Förfrågan från köpprocessen  //Vi extraherar N-parametern från den interna köprocessen  else if received signal = answer {  Extract N from signal answer;  write(outfile, N);  } |

Föregående tillstånd uppe

Nästa tillstånd nere

