Simulation

Simulation is experiments with a model of a system

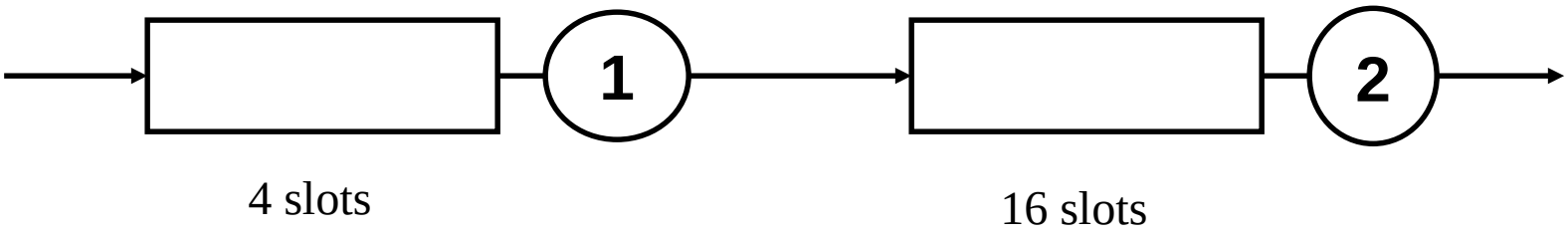
- Event scheduling method
- Process interaction method

Event scheduling approach

What is needed:

- A state description
- Events
- Rules telling what will happen when an event occurs
- Parameters

A more complicated example



We want to find

- the mean number of customers in queueing system 1 and 2
- the probability that a customer is rejected when it arrives to queueing system 1

State description

N1 = number of customers in queueing system 1

N2 = number of customers in queueing system 2

Measuring variables:

- NoOfArrivals (is just what you think!)
- NoRejected (is just what you think!)

This is not state variables in a strict sense but they also have to be updated at certain events!

Events needed

- ArrivalTo1
- DepartureFrom1
- DepartureFrom2
- Measurement

Rule for ArrivalTo1

```
void RuleArrivalTo1(){
   NoOfArrivals++;
   If (N1 < 4)
      N1++;
   else
      NoRejected++;
   If (N1 == 1)
      InsertEvent(DepartureFrom1, time + 0.2);
   InsertEvent(ArrivalTo1, time + nextArrival());
}
```

Rule for DepartureFrom1

```
void RuleDepartureFrom1{
    N1--;
    if (N2 < 16)
        N2++;
    if (N2 == 1)
        InsertEvent(DepartureFrom2, time + 0.1);
    if (N1 >) 0 then
        InsertEvent(DepartureFrom1, time + 0.2);
}
```

Rule for DepartureFrom2

```
void RuleDepartureFrom2{
   N2--;
   if (N2 > 0)
      InsertEvent(DepartureFrom2, time + 0.1);
}
```

Rule for Measurement

```
void RuleMeasurement{
   write(file1, N1);
   write(file2, N2);
   InsertEvent(Measurement, time +
     NextMeasurement());
}
```

Another example

Assume that we want to measure the probability that a customer spends more than 5 seconds in the system.

Then it is not enough to keep track of the number of customers in the queueing system!

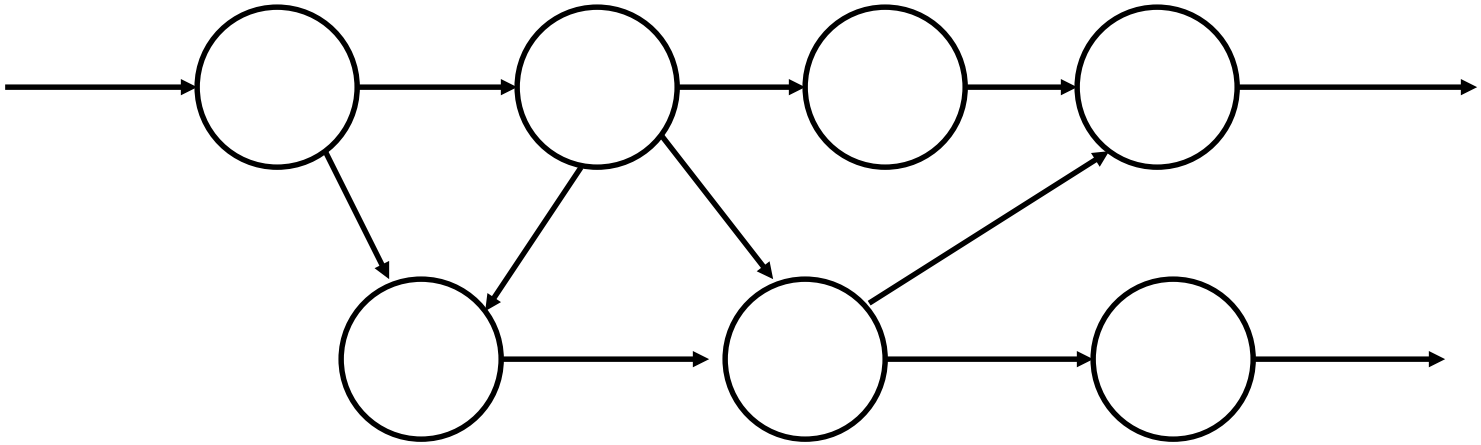
Events here are Arrival and Departure.

The state of this system

In this case the state can be a list where we can store customers and mark them with their arrival time:

Can be implemented by a double linked list or a vector

Drawback of event scheduling



Assume that we have a complicated network with many nodes. The network can model e.g. a computer network, material flow or luggage handling. The nodes are similar.

Drawbacks

- Many different events or events with attributes are needed
- It is difficult to change the system, a change in one of the nodes affects the programs global variables and rules
- It is more natural to think of such a problem as entities flowing through the network, than to think about events

What we would like

We would like to

- create a template for the nodes and customers
- That when the program executes, instances of the nodes and customers are created
- set parameters to the instances when they are created

The solution

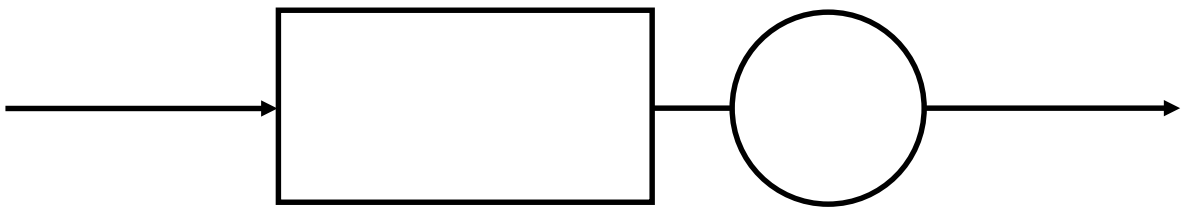
One way of solving this is the

process interaction method

Processes in simulation

- In simulation a *process* is something that does something
- A process has some *internal state*
- Processes communicate by sending *signals* to each other
- Signals have a name and can carry information
- When a signal arrives to a process some *activity* is trigged
- During an activity the state of the receiving process might be changed and signals may be sent
- When a signal is sent the sender assigns it an *arrival time*

An example



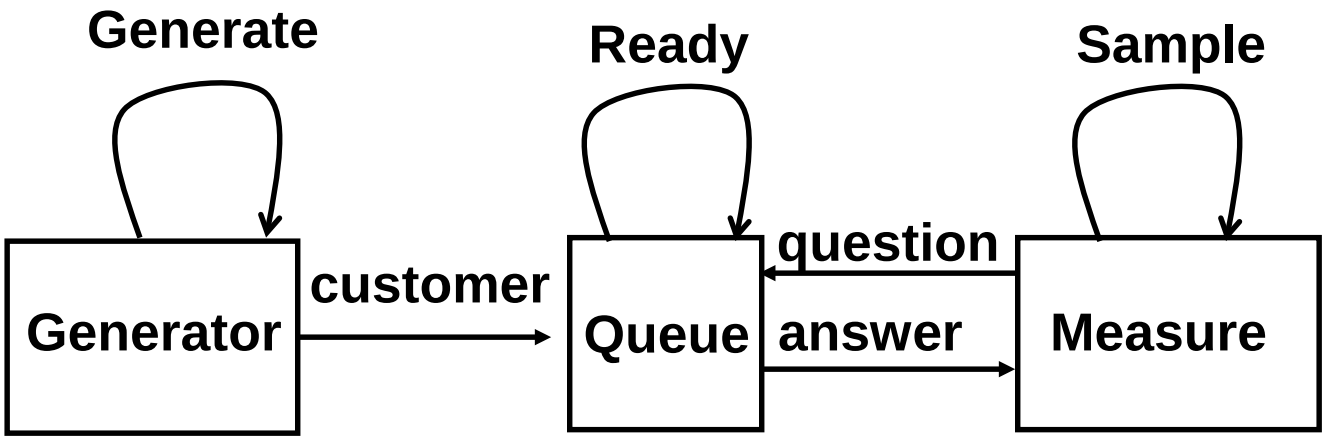
Assume that we want to describe a queueing system with the process interaction method.

The processes we need

One process

- representing the queueing system
- generating customers
- measuring the number of customers in the queueing system

The processes and signals



Generate, ***Ready*** and ***Sample*** are delayed

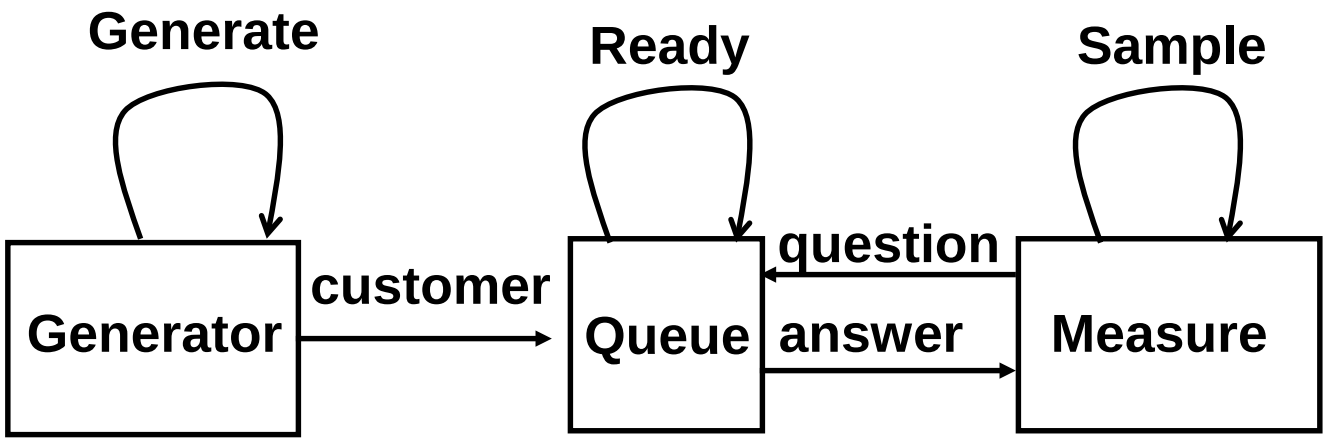
answer has a parameter, the number of customers.

The internal state of the processes

- **Generator**: no internal state needed
- **Queue**: N = number of customers
- **Measure**: no internal state needed

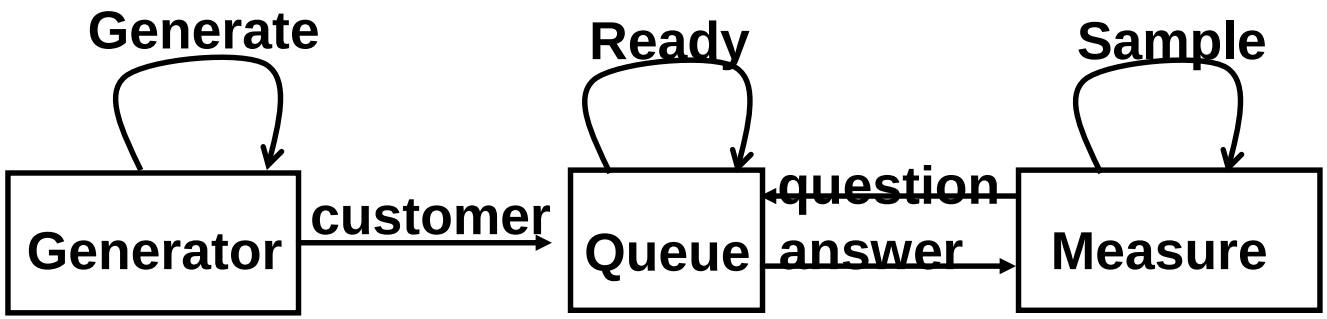
Activity of Generator

```
if received signal = generate {
   SendSignal(customer, Queue, time);
   SendSignal(generate, Generator, time + Exp(4));
}
```



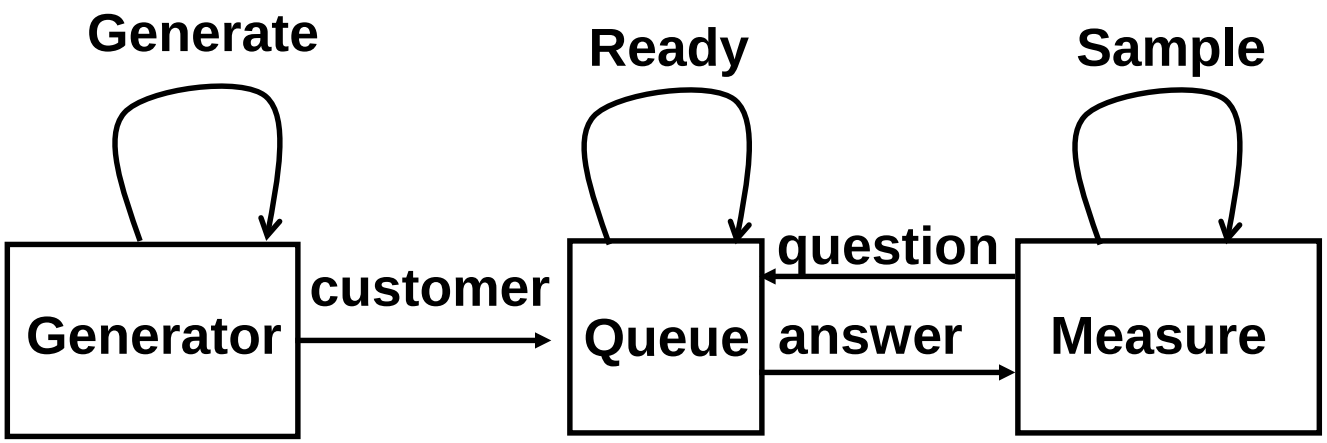
Activity of Queue

```
if received signal = customer{
   N := N + 1;
   if N = 1 then
      SendSignal(ready, Queue, time + Exp(2));
}
else if received signal = ready{
   N := N – 1;
   if N > 0 then
      SendSignal(ready, Queue, time + Exp(2));
}
else if received signal =  question{
   SendSignal(answer(N), Measure, time);
}
```



Activity of Measure

```
If received signal = sample {
   SendSignal(question, Queue, time);
   SendSignal(sample, Measure, time + Exp(10));
}
else if received signal = answer {
   Extract N from signal answer;
   write(outfile, N);
}
```



Some problems we must solve

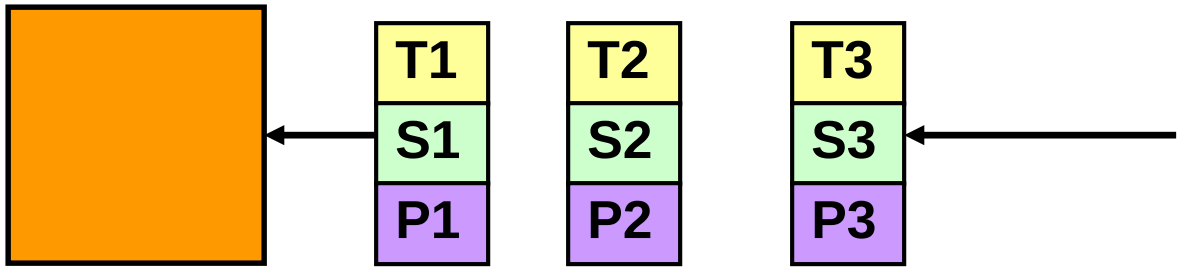
- How to keep track of time in the system
- How to make sure that signals arrive at the right time

Observe that it is not a question of real time!

Time is just updated when a signal arrives. It does not have any values in between.

Signal list

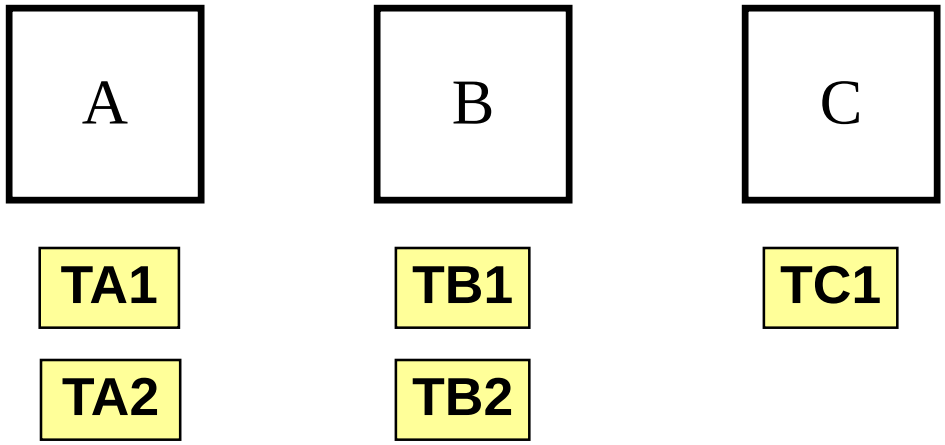
Each process has a signal list. It is very similar to the event list in the event scheduling method.



T_i = arrival time of signal
S_i = what kind of signal this is (the name of the signal)
P_i = parameters of the signals (if any)
$T_1 < T_2 < T_3 <$ etc

Process list

Processes with signals in their signal lists are organized in a process list. Only the arrival times of the signals are shown here.



The process list is sorted so that $\mathbf{TA_1 < TB_1 < TC_1 < etc}$

How the process interaction method works

1. **Remove the first process from the process list (call it A)**
2. **Remove the first signal in A:s signal list**
3. **Process the activities**
4. **If there are any signals left in A:s signal list, sort it into the process list again**
5. **If simulation shall continue, go to 1**

What to do when a process gets a signal

Assume that process B gets a signal.

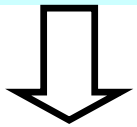
✓ Sort the signal into process B:s signal list.

- If the signal list was empty before the signal arrived, B shall be sorted into the process list.
- If the signal list was not empty, B is already in the process list.
  - If the signal is put first in B:s signal list, B might have to change its place in the process list.

An example, the queuing system (1)

Time = 0
Generator: (3,arrival)
Measure: (10, sample)

Queue: [N=0] () Queue is not in the process list!



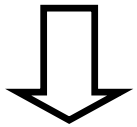
Time = 3

Queue: [N=0] (3,customer)
Measure: (10, sample)
Generator: (11, generate)

An example, the queuing system (2)

Time = 3

Queue: [N=0] (3,customer)
Measure: (10, sample)
Generator: (11, generate)



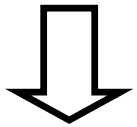
Time = 3

Measure: (10, sample)
Generator: (11, generate)
Queue: [N=1] (12,ready)

An example, the queuing system (3)

Time = 3

Measure: (10, sample)
Generator: (11, generate)
Queue: [N=1] (12,ready)



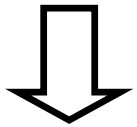
Time = 10

Queue: [N=1](10,question)(12,ready)
Generator: (11, generate)
Measure: (20, sample)

An example, the queuing system (4)

Time = 10

Queue: [N=1](10,question) (12,ready)
Generator: (11, generate)
Measure: (20, sample)



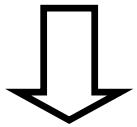
Time = 10

Measure: (10, answer) (20, sample)
Generator: (11, generate)
Queue: [N=1](12,ready)

An example, the queuing system (5)

Time = 10

Measure: (10, answer) (20, sample)
Generator: (11, generate)
Queue: [N=1](12,ready)



Time = 10

Generator: (11, generate)
Queue: [N=1](12,ready)
Measure: (20, sample)

The steps in constructing a process interaction simulation program

Which

- processes are needed?
- variables are needed to describe the state of the processes?
- signals are needed?
- information (besides its name) shall a signal contain?
- activity shall occur when a signal arrives at a process?

When these questions are answered, it is not difficult to write a process interaction simulation program!

Time spent thinking on these questions
will save a lot of time later!

A further wish

- We would like to define process types, e.g. generator and queue. When we start a program we would like to create as many instances of these types as we need.

- In this way we can create a library of processes that can be reused. This is one more advantage of the process interaction approach.

Only one signal list

- It is possible to use just one signal list in a program
- In that case the implementation of a process interaction simulation program is very similar to a event scheduling simulation program