Stochastic Simulation
Discrete simulation/event-by-event

Bo Friis Nielsen

Institute of Mathematical Modelling

Technical University of Denmark

2800 Kgs. Lyngby – Denmark

Email: bfni@dtu.dk

Discrete event simulation

- Continuous but asynchronous time
- Systems with discrete state-variables
  - Inventory systems
  - Communication systems
  - Traffic systems - (simple models)
- even-by-event principle

Elements of a discrete simulation language/program

- Real time clock
- State variables
- Event list(s)
- Statistics

The event-by-event principle

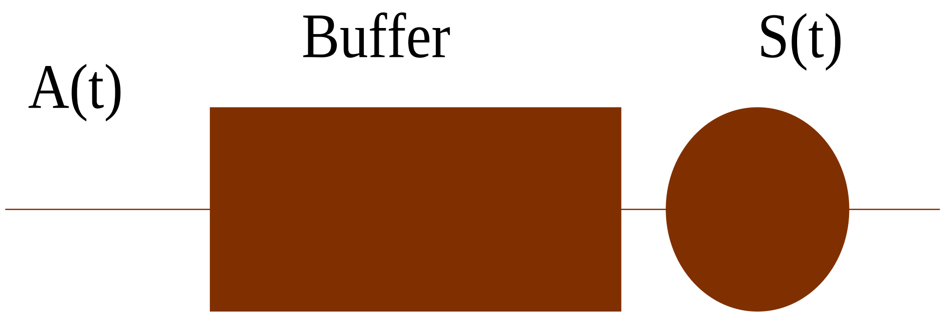
- Advance clock to next event to occur
- Invoke relevant event handling routine
  - collect statistics
  - Update system variables
- Generate and schedule future events - insert in event list(s)
- return to top

Analysing steady-state behaviour

- Burn-in/initialisation period
  - ◇ Typically this has to be determined experimentally
- Confidence intervals/variance estimated from sub-samples

Queueing systems

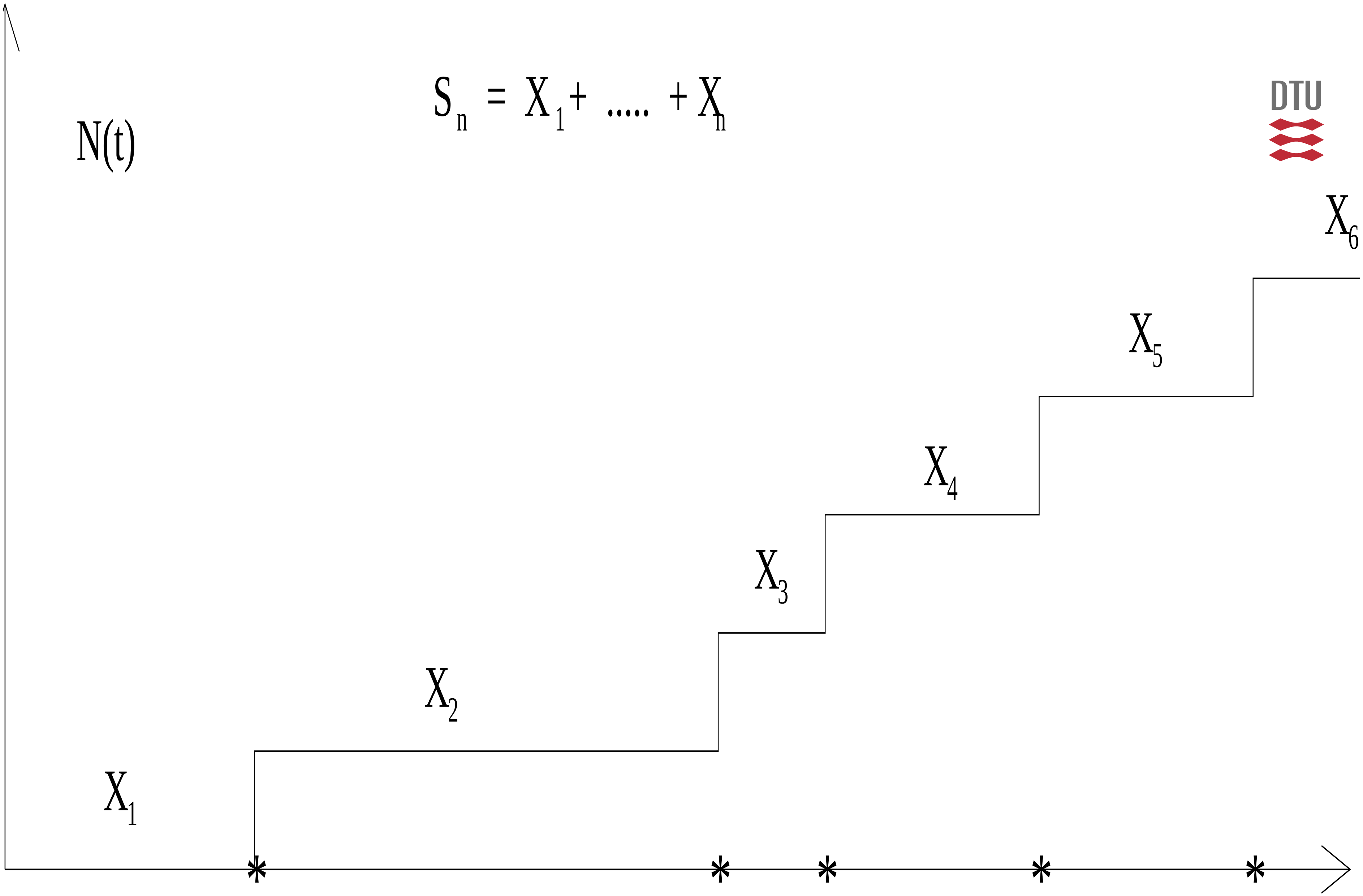
- Arrival process
- Service time distribution(s)
- Service unit(s)
- Priorities
- Queueing discipline



- $A(t)$ - Arrival process
- $S(t)$ - Service process (service time distribution)
- Finite or infinite waiting room
- One or many serververs
- Kendall notation: $A(t)/S(t)/N/K$
  - N - number of servers
  - K - room in system (sometime K only relates to waiting room)

Performance measures

- Waiting time distribution
  - ◇ Mean
  - ◇ Variance
  - ◇ Quantiles
- Blocking probabilities
- Utilisation of equipment (servers)
- Queue length distribution



N(t)
S n = X 1 + ... + X n
X1
X2
X3
X4
X5
X6

Poisson process

- Independently exponentially distributed intervals
- Poisson distributed number of events in an interval. Number of events in non-overlapping intervals independent

$$P(X_i \leq t) = 1 - e^{-\lambda t} \qquad N(t) \sim P(\lambda t) \Leftrightarrow P(N(t) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$$

- If the intervals are independently but generally distributed we call the process a renewal process

Sub-samples - precision of estimate

- We need sub-samples in order to investigate the precision of the estimate.
- The sub-samples should be independent if possible
- For independent subsamples the standard deviation of our estimate will be proportional to $\sqrt{n}^{-1}$

Confidence limits based on sub-samples

- We want to estimate some quantity θ
- We obtain n different (independent) estimates $\hat{\theta}_i$.
- The central limit theorem motivates us to construct the following confidence interval:

$$\bar{\theta} = \frac{\sum_{i=1}^{n} \hat{\theta}_i}{n} \qquad S_\theta^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} \hat{\theta}_i^2 - n\bar{\theta}^2\right)$$

$$\left[\bar{\theta} + \frac{S_\theta}{\sqrt{n}} t_{\frac{\alpha}{2}}(n-1); \bar{\theta} + \frac{S_\theta}{\sqrt{n}} t_{\frac{1-\alpha}{2}}(n-1)\right]$$

General statistical analysis

- More generally we can apply any statistical technique
- In the planning phase - experimental design
- In the analysis phase
  - Analysis of variance
  - Time-series analysis
  - ⋮

Rødby Puttgarden Simulation study

- Access to harbour facilities
- A number of rules regarding
  - ◇ The harbour channel
  - ◇ The ferry berths
- Data for
  - ◇ Sailing times
  - ◇ unload/load times

The system events

- arrive_harbour;
- depart_channel
- loaded
- ready_sail
- arrive_channel
- arrive_berth

Global (ressource) variables

```
short channel[2],berth[2][4];
```

Ferry data structures

```
class ferry_type {

public:
  short type;
  short id;
  short status;
  short event;
  short harbour;
  short berth;
  time_type event_time;
  time_type scheduled_time;
  ferry_type * previous;
  ferry_type * next;
};
```

Main programme - initialisation

```
void main()
{
  ferry_type * ferry;

  time.minutes=0;
  time.hours=0;
  event_list = 0;
  Initialization();
  Print_ferries(event_list);
```

Main programme simulation loop

```
while (time.hours<100)
{
  ferry = event_list;
  time = ferry->event_time;
  event_list = event_list->next;
  if (event_list != 0) event_list->previous =0;
  switch(ferry->event)
  {
    case arrive_harbour : Arrive_harbour(ferry); break;
    case depart_channel : Depart_channel(ferry); break;
    case loaded         : Loaded(ferry);         break;
    case ready_sail     : Ready_sail(ferry);     break;
    case arrive_channel : Arrive_channel(ferry); break;
    case arrive_berth   : Arrive_berth(ferry); break;
    default             : break;
  } /* End switch */
} /* End main loop */
Print_statistics();
```

Sample event procedure

```
void Arrive_harbour(ferry_type * ferry)
{
  if ((Request_channel(ferry)>0) &&
      (Request_berth(ferry)>0))
  {
    ferry->event = arrive_channel;
    ferry->event_time = time;
    Insert_in_event_list(ferry);
  }
  else
    Wait_for_arrive(ferry);
}
```

Another event procedure

```
void Depart_channel(ferry_type * ferry)
{
  channel[ferry->harbour] = vacant;
  ferry->event = arrive_harbour;
  ferry->event_time = time + Sailing_time(ferry);
  Check_waiting_ferries(ferry->harbour);
  ferry->harbour = New_harbour(ferry->harbour);
  Insert_in_event_list(ferry);
}
```

Exercise 4

- Write a discrete event simulation program for a blocking system, i.e. a system with n service units and no waiting room.
- The arrival process is modelled as a Poisson process.
- Choose first the service time distribution as exponential.
- Record the fraction of blocked customers, and a confidence interval for this fraction..
- The programme should take the offered traffic and the number of service units as input parameters.

Example: n = 10, mean service time = 8 time units, mean time between customers = 1 time unit (corresponding to an offered traffic of 8 erlang), 10 x 10.000 customers.

Exercise 4 - continued

- In the above example substitute the arrival process with a renewal process with 1) Erlang distributed inter arrival times 2) hyper exponential inter arrival times. The Erlang distribution should have a mean of 1, the parameters for the hyper exponential distribution should be $p_1 = 0.8, \lambda_1 = 0.8333, p_2 = 0.2, \lambda_2 = 5.0$.
- Finally experiment with different service time distributions. Suggestions are constant service timeand Pareto distributed service times,for Pareto will $k = 1.05$ and $k = 2.05$ be interesting choices. It is recommended that the service time distribution has the same mean (8).
- Make the experiment with a distribution of (your own) choice. Remember that the distribution should take only non-negative values.

Exercise 4 - exact solution

- With arrival intensity λ and mean service time s
- Define $A = \lambda s$
- Erlangs B-formula

$$B = P(n) = \frac{\frac{A^n}{n!}}{\sum_{i=0}^{n} \frac{A^i}{i!}}$$

- Valid for all service time distributions
- But arrival process has to be a Poisson process