Network Protocol Design and Evaluation

07 - Simulation, Part I

Stefan Rührup

University of Freiburg
Computer Networks and Telematics
Summer 2009
Overview

‣ In the last chapters:
  • Formal Specification, Validation, Design Techniques
  • Implementation → Software Engineering, Lab Courses

‣ In this chapter:
  • Performance evaluation by simulation
  • Simulation models
After the design phase...

- Implementation and Test?
  - A good idea, but testing in a real environment requires a lot of effort.
  - You might want to start with a prototype to get some more insights.

- Alternative evaluation methods?
Evaluation

Methods of performance evaluation:

- Experiments: Measuring performance in a concrete example in a real environment.
- Simulation: Numerical evaluation of a system model in an artificial environment.
- Analysis: Describing properties of a mathematical abstraction of the system.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Evaluation Methods

Real World Experiment

Wingtip vortices
*Image Credit: NASA*

Physical Model

Model in a wind tunnel
*Image Credit: NASA*

Mathematical Model

Computational Fluid Dynamics
*Image Credit: NASA*
Mathematical System Models

- **Static vs. dynamic**
  - Static models cover a certain fixed state of a system. State changes are not considered.
  - Dynamic models reflect the system’s state changes over time.
  - The time model can be continuous or discrete.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Continuous vs. Discrete Models

- **Time-continuous models**
  - State variables are continuous functions over time, e.g. description of variable changes by differential equations.

- **Time-discrete models**
  - State changes happen only at discrete time points (state variables do not change in between).
  - We call these time points **events**.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
System and Models

- The choice of the type of model depends on objectives and feasibility!
- Continuous systems may be described by discrete models and vice versa. Examples:
  - Voice communication (continuous system) may be described by a discrete model if digital samples are transmitted.
  - Internet traffic (discrete system with packet transmissions as events) can be described by a continuous model, if the large-scale behaviour is of interest.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Deterministic vs. Stochastic Models

- **Deterministic models**
  - The sequence of state changes depend on the initial state can be completely described

- **Stochastic models**
  - The sequence of state changes depend on random events

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
System Models

Physical

Mathematical

Static

Dynamic

Continuous

Deterministic

Stochastic

Discrete

Deterministic

Stochastic

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Evaluation Goals

- The goal of experimental or simulative studies is the evaluation of some system properties
  - Gain insight in system behaviour
  - Get performance estimations
  - Use results to improve the design
  - Reduce cost

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Parameters and Metrics

• Most systems have a set of parameters that determine their behaviour.

• An evaluation tries to characterize a system by a set of metrics.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulation of Discrete Models

- We consider the simulation of communication protocols
- Models are usually dynamic, time-discrete and stochastic

- **Generic procedure:**
  1. Implement the model for system behaviour
  2. Define parameters
  3. Run the simulation
  4. Observe metrics
  5. Evaluate results (this will raise more questions...)

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
A generic simulation model

- ... for discrete event simulation.
- Elements:
  - Simulation clock
  - Event list (sorted w.r.t. time)

- Next-event time advance algorithm:
  - Initialize simulation clock to 0; Initialize future event list.
  - Repeat
    - Advance simulation clock to next event in list
    - Update system state and insert new events in list
  - Until event list empty or simulation time exceeded

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulation example: Are you being served?

- Simulating a queue:
  - Customers wait in a queue to be served.
  - It requires some time to serve each customer.
  - How long do you have to wait?
Modeling a Queue (1)

- **Simulation model:**
  - There is one counter
  - Customers appear at certain time points
  - The service itself requires some time

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Modeling a Queue (2)

- **Parameters:**
  - Patterns of customer arrival and service times:
    - When do customers arrive?
    - How long does it take to serve each customer?
  - Stochastic model: Inter-arrival time as random variable
    (usually assumed to be exponentially distributed)

- **Metrics:**
  - Waiting time (average, maximum)
  - Queue length (average, maximum)
  - Server utilization

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulating a Queue (2)

What about service times?
Simulating a Queue (3)

Arrival of...

\[ t_0 \quad t_1 \quad t_2 \quad t_3 \]

Service times: A B C

Arrival of...

\[ t_0 \quad t_1 \quad t_2 \quad t_3 \]
Measuring Waiting Time

Arrival of...

\[ \text{Delay} \quad d_0 \quad d_1 \quad d_2 \]

\[ \text{Average waiting time: } \frac{1}{n} \sum d_i \]

\[ (n = \text{number of customers}) \]

\textit{Discrete-time metric}: Average taken over a discrete set of numbers

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Measuring Queue Length

**Average queue length:**
Area under the red curve / total time

How to compute: Maintain total area in a variable. Add area since last event when processing a new event.

*Continuous-time metric:* Average taken continuously over time

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Measuring Server Utilization

Arrival of... A B C

t_0 t_1 t_2 t_3 t_{end} time

busy time total time

Server utilization: Busy time / total time
On the Meaning of Measurements (1)

- How to interpret these measurements?
- In the previous example:
  - Per-customer delays are measured in one particular simulation run
  - Different runs, different delays → different results

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
On the Meaning of Measurements (2)

› Why use aggregated information?
› Aggregated information (here: average) gives a concise description of system characteristics
› Why not consider distributions instead of measured averages?
› In the previous example:
  • The first customer does *never* have to wait (d₀=0), which is not true for the following ones
  • This real behaviour is not covered by a distribution

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
On the Meaning of Measurements (3)

- Goal: Extract the “truly typical” behaviour of the model
- Simulation runs give only an estimator for this behaviour

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Lessons learned

- Simulation means to model a system and run it in an artificial environment

- **Parameters** select system behaviour

- **Metrics** characterize a system

- Discrete event models do not only describe discrete time systems.

- Stochastic models are often used when mathematical models are intractable.
Implementing a Simulation Model

- **Overview:**
  - Next-event time advance algorithm
  - Maintaining the future event set
  - Adding statistics
  - Randomness
  - Object-oriented implementation
  - Race conditions
Next-event time advance algorithm

- **Basic algorithm**

  initialize;

  while (stopping condition is false) {
    get next event from future event set
    advance time to this event
    process event
    generate new event(s) and add them to the future event set
  }

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulating the Queue (1)

- **State of the model:**
  - Simulation time
  - Queue for storing the tasks (FIFO)
  - Server state: idle or busy
  - Events: arrival and departure.
    Variables:
    - Time of next arrival
    - Time of next departure

- Arrivals and departures are modeled as a Poisson process.
  Inter-arrival times follow an exponential distribution

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulating the Queue (2)

- **Processing events:**
  - on arrival: add task to queue, schedule departure
  - on departure: remove task from queue

- **Initialization:**
  - generate first arrival time

- **Setting the server state**
  - If a task arrives, the server is busy
    (further tasks have to wait)
  - If the queue is empty after processing, the server is idle

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulating the Queue (3)

server_state = IDLE;
double sim_time = 0.0;
double next_departure = HUGE_VAL; ← set to infinity
double next_arrival = random.poisson(mean_arrival); ← first event

while (sim_time < T_MAX) {
    if (next_arrival < next_departure) { ← next event: arrival
        sim_time = next_arrival; ← advance clock
        if (server_state == IDLE) {
            server_state = BUSY;
            next_departure = sim_time + random.exp(mean_processing);
        } else {
            queue.push( new task(sim_time) );
        }
        next_arrival = sim_time + random.exp(mean_arrival);
    } else { ← next event: departure
        sim_time = next_departure;
        if ( queue.empty() ) {
            server_state = IDLE;
            next_departure = HUGE_VAL;
        } else {
            task t = queue.pop();
            next_departure = sim_time + random.exp(mean_processing);
        }
    }
}
Simulating the Queue (4)

- Next-event time advance:

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulating the Queue (5)

- Key techniques used here:
  - When an arrival event is processed, the next event of the same type is generated (event occurrence is modeled by a Poisson process)
  - Departure events are “poisoned” if there is no waiting task (by setting its time to infinity)

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Randomness

- Random numbers are essential to stochastic models

- There are sources of true randomness
  - difficult to produce, difficult to re-produce
  - Reproducible simulation results are desired!

- Therefore, **pseudo-random** numbers are used, which can be generated **deterministically**

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Types of RNGs

- Pseudo random number generators (PRNGs) generate sequences of numbers
- PRNGs can only produce a finite quantity of different numbers and sequences of finite length (until numbers are repeated)

- There are various PRNGs. **Which are good, which are bad?**
  - Good RNGs produce every possible number (or a large fraction) of the value range
  - Good RNGs have a long period
  - Bad RNGs are sensitive to seed selection
Types of PRNGs: LCG

- Linear congruential generator (LCG):
  - \( x_{n+1} = (ax_n + c) \mod m \)
  - Parameters:
    - \( m \): modulus, \( m > 0 \)
    - \( a \): multiplier, \( 0 < a < m \)
    - \( c \): increment, \( 0 \leq c < m \)
    - \( x_0 \): start value or “seed”, \( 0 \leq x_0 < m \)

- **Example:** \( x_{n+1} = (3x_n + 5) \mod 8 \), \( x_0 = 5 \)
  - Sequence: 5, 4, 1, 0, 5, 4, 1, 0, 5, 4, 1, 0,...

- Easy implementation, but short period and sensitive to parameters
- Used in glibc, ANSI C, MS C++
Types of PRNGs: LFSR

- Linear feedback shift register (LFSR):

  - Period depends on the feedback function
  - There are tables with maximum cycle-length polynomials

Feedback polynomial: \( x^{16} + x^{14} + x^{13} + x^{11} + 1 \)
Types of PRNGs: LFG

- Lagged Fibonacci generator:
  - Fibonacci sequence: $F_n = F_{n-1} + F_{n-2}$, $F_0 = 0$, $F_1 = 1$
  - Generalized form: $F_n = (F_{n-j} \ op \ F_{n-k}) \mod m$, $0 < j < k$
    $\ op = \text{binary operator}$
  - Good choices for $(j, k)$ are e.g. $(7, 10), (5, 17), (6, 31)$.
    (see D. Knuth, The Art of Computer Programming, Vol. 2)

- Longer period than LCGs, but sensitive to initialization
- Used in Boost
Types of PRNGs: Mersenne Twister

- **Mersenne Twister** (Matsumoto, Nishimura 1997)
  - Twisted generalized feedback shift register
  - Very long period
  - High order of dimensional equidistribution (low correlation between successive numbers)
  - More complicated to implement
    (look for standard implementation of MT19937)

- Implemented as part of GSL (GNU scientific library)
- **Alternative**: Complementary Multiply-with-carry generator
## PRNGs Overview

<table>
<thead>
<tr>
<th>PRNG</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCG (mod m)</td>
<td>max. m</td>
</tr>
<tr>
<td>LFG (mod m, operator = addition)</td>
<td>max. ((2^k-1)\times(m/2))</td>
</tr>
<tr>
<td>LFSR (n bits)</td>
<td>max. (2^n-1)</td>
</tr>
<tr>
<td>Mersenne Twister MT19937 (32 bit)</td>
<td>(2^{19937}-1 = 4.3 \times 10^{6001})</td>
</tr>
<tr>
<td>Complementary Multiply-with-carry</td>
<td>(10^{410928})</td>
</tr>
</tbody>
</table>
RNGs and Distributions (1)

- PRNGs produce a sequence of (hopefully) uniform distributed numbers.
- Such sequence can be used to obtain other distributions
- Example for generating Poisson-distributed numbers:

```c
int rand_poisson(double lambda) {
    double L = e^(-lambda);
    int k = 0;
    double p = 1;
    repeat
        k = k + 1;
        r = uniform random number in [0,1];
        p = p * r;
    until (p <= L)
    return k - 1;
}
```

RNGs and Distributions (2)

- Generating random numbers an arbitrary distribution with density function \( f \) and cumulative distribution function \( F \)
  - **Inverse transform sampling**
    1. Generate a uniformly distributed random number \( u \)
    2. Compute \( x \) such that \( F(x) = u \)
    3. return \( x \)
  - **Rejection sampling**
    Choose a function \( g \) such that \( f(x) \leq c \cdot g(x) \) for all \( x \in \mathbb{R} \) and a constant \( c \)
    1. Generate a uniformly distributed random number \( u \)
    2. Sample \( x \) randomly from \( c \cdot g(x) \)
    3. If \( u < f(x) / c \cdot g(x) \) return \( x \); otherwise goto step 1
RNGs and Distributions (2)

- Generating random numbers for arbitrary distributions
  - Inverse transform sampling
  - Rejection sampling

1. Generate a uniform

- Rejection sampling
Seed selection

- PRNGs produce sequences deterministically: Same seed - same PRN sequence
- Use seeds as simulation parameters
- Don’t use 0 (e.g. LCG with c=0 gets stuck)
- Avoid even numbers (e.g. LCG with c=0 and even m produces fewer different numbers)

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Multiple RNGs (1)

- In the example, the same RNG is used for inter-arrival and processing times

```c
while (sim_time < T_MAX) {
    if (next_arrival < next_departure) {
        sim_time = next_arrival;
        if (server_state == IDLE) {
            server_state = BUSY;
            next_departure = sim_time + random.exp(mean_processing);
        } else {
            queue.push( new task(sim_time) );
        }
        next_arrival = sim_time + random.exp(mean_arrival);
    } else {
        sim_time = next_departure;
        if ( queue.empty() ) {
            server_state = IDLE;
            next_departure = HUGE_VAL;
        } else {
            task t = queue.pop();
            next_departure = sim_time + random.exp(mean_processing);
        }
    }
}
```
Multiple RNGs (2)

- Using the same PRNG can lead to correlations between inter-arrival and processing times.

- Here, arrival and processing times are independent random variables.

- Initialization with different parameters should be possible

- **General recommendations:**
  - Use different PRNGs for different random variables
  - Use different seeds!
Multiple RNGs (3)

- Overlapping sequences should be avoided:

```
  seed 1   seed 2
```

```
RNG 1  sequence 2
RNG 2  sequence 1
```

- More recommendations:
  - Ensure that seeds separate the sequences
  - Limit simulation length

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Adding Statistics (1)

```c
double last_event_time = 0.0;
waiting_time_total = 0.0;

while (sim_time < T_MAX) {
    if (next_arrival < next_departure) sim_time = next_arrival;
    else sim_time = next_departure;
    update_statistics();
    if (next_arrival < next_departure) {
        if (server_state == IDLE) {
            server_state = BUSY;
            next_departure = sim_time + rng2.exp(mean_processing);
        } else {
            queue.push( new task(sim_time) );
        }
        next_arrival = sim_time + rng1.exp(mean_arrival);
    } else {
        if ( queue.empty() ) {
            server_state = IDLE;
            next_departure = HUGE_VAL;
        } else {
            task t = queue.pop();
            next_departure = sim_time + rng2.exp(mean_processing);
            waiting_time_total = sim_time - t.arrival_time;
        }
    }
    last_event_time = sim_time;
}
```
Adding Statistics (2)

- Order of execution: Advance clock, record statistics, process events (state changes)

```c
void update_statistics() {
  double time_since_last_event = sim_time - last_event_time;
  cumulated_queue_length += queue.length() * time_since_last_event;
  if (server_state == BUSY)
    busy_time_total += time_since_last_event;
}

// at the end of the simulation:
avg_queue_length = cumulated_queue_length / sim_time;
avg_utilization = busy_time_total / sim_time;
```
Recording statistics

- Simulation parameters for the queueing example:
  - RNG parameters: seeds and mean values (mean inter-arrival time A and mean service/processing time S)
  - Simulation duration (time limit or max number of tasks)

Results for 1000 tasks, A = 1, S=0.5

<table>
<thead>
<tr>
<th>Seed A</th>
<th>Seed S</th>
<th>Avg. Utilization</th>
<th>Avg. Queue Length</th>
<th>Avg. Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>17</td>
<td>0.534</td>
<td>0.572</td>
<td>0.563</td>
</tr>
<tr>
<td>25</td>
<td>89</td>
<td>0.496</td>
<td>0.478</td>
<td>0.479</td>
</tr>
<tr>
<td>167</td>
<td>11</td>
<td>0.505</td>
<td>0.458</td>
<td>0.453</td>
</tr>
<tr>
<td>235</td>
<td>21</td>
<td>0.506</td>
<td>0.451</td>
<td>0.435</td>
</tr>
</tbody>
</table>

Simulations performed with example program in:
[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Processing statistics

- Average over different simulation runs:

<table>
<thead>
<tr>
<th>Seed A</th>
<th>Seed S</th>
<th>Avg. Utilization</th>
<th>Avg. Queue Length</th>
<th>Avg. Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>17</td>
<td>0.534</td>
<td>0.572</td>
<td>0.563</td>
</tr>
<tr>
<td>25</td>
<td>89</td>
<td>0.496</td>
<td>0.478</td>
<td>0.479</td>
</tr>
<tr>
<td>167</td>
<td>11</td>
<td>0.505</td>
<td>0.458</td>
<td>0.453</td>
</tr>
<tr>
<td>235</td>
<td>21</td>
<td>0.506</td>
<td>0.451</td>
<td>0.435</td>
</tr>
</tbody>
</table>

Average: 0.510  0.490  0.483
Standard error: 0.008  0.028  0.028

Simulations performed with example program in:
[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
The Impact of Simulation Time (1)

- Example: Simulating a busy server
  Mean inter-arrival time $A = 1$, mean service time $S = 0.9$
  Seeds $A,S = 23,17$
- Analytical results available (M/M/1 queue)

<table>
<thead>
<tr>
<th>Number of tasks</th>
<th>Avg. Utilization</th>
<th>Avg. Queue Length</th>
<th>Avg. Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.563</td>
<td>0.490</td>
<td>0.585</td>
</tr>
<tr>
<td>100</td>
<td>0.849</td>
<td>1.885</td>
<td>2.093</td>
</tr>
<tr>
<td>1000</td>
<td>0.957</td>
<td>9.965</td>
<td>9.839</td>
</tr>
<tr>
<td>10000</td>
<td>0.913</td>
<td>8.459</td>
<td>8.496</td>
</tr>
<tr>
<td><strong>Analytical:</strong></td>
<td><strong>0.9</strong></td>
<td><strong>8.1</strong></td>
<td><strong>8.1</strong></td>
</tr>
</tbody>
</table>

Simulations performed with example program in:
[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
The Impact of Simulation Time (2)

- For short simulations (small number of tasks) the results highly deviate from analytical values.
- Long simulation runs seem to produce better results.
- What is the reason for this behaviour?
- We observe the queue length over time...
The Impact of Simulation Time (3)

Queue length statistics

Number of tasks in queue vs. Simulation time

analytical avg.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
The Impact of Simulation Time (4)

- At the beginning the queue is empty
- It needs some time to fill the queue, a high number of waiting tasks is unlikely
- This initial phase \((\text{initial transient})\) should be removed, if the typical steady-state behaviour is to be observed

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Object-oriented design (1)

- Separate Modules
  - Objects which are simulated
  - Event dispatcher to handle events
  - Load generator
- Modules are extended (communicating) FSMs. They communicate only by message exchange.
- Strict separation of simulation engine from problem-specific modules and events
Object-oriented design (2)

- Data structure for the future event set
  - Priority queue
  - Events are sorted according to their activation time, next event = minimum key element
  - Which data structures are efficient? Heap structures, e.g. Fibonacci heaps, provide quick access to the minimum element and efficient insertion
- Implementation: exercise!

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Avoiding Race Conditions

- Handling simultaneous event arrival:
  - Priorities for different event classes
  - Maintaining the order (according to creation)
  - Testing conditions and subsequent state changes as atomic sequence

- Determinism should always be ensured!
Network Simulation

- A typical simulation setup:
- Protocols are Communicating FSMs
- They are fed by a load generator
- The simulation engine passes messages between the modules and triggers timer events
  - Core modules: Event dispatcher and event queue (future event set)
Simulation Model (1)

- **System Model:**
  - describes the composition of the whole system into modules (and submodules)
    - Module parameters: queue size, processing delay
  - Modules communicate by message exchange over links
    - Link parameters: delay, bandwidth limitation

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Simulation Model (2)

- **Load Model:**
  - describes the pattern of requests made to the system, e.g. how often are messages injected by an application?

- **Fault Model:**
  - describes the deviation from normal behaviour, e.g. module failures, lossy links

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
A Typical System Model

Host A
- Appl. Layer: Traffic Generator
- MyProtocol
- Lower Layer Abstraction

Host B
- Appl. Layer: Traffic Generator
- MyProtocol
- Lower Layer Abstraction

Transmission Medium
Message Passing in Simulation

Extended FSM

Module A

send (msg, channel)

defines sender/receiver and delay

channel

Module B

Event Dispatcher

msg [sim_time + delay]

Future Event Set

calls scheduling function of the event dispatcher

simulation engine
Message Passing in Simulation

Module A

channel

Module B

handleEvent(msg)

calls message handling function of Module B

Event Dispatcher

next event: msg

Future Event Set
Generating Load

- Sources of load patterns
  - Traces (recorded data)
  - Empirical distributions
  - Models for arrival processes
Traces

- Using recorded data of real systems in simulations
  - recorded arrival and service times of a queueing system, recorded packet loss, mobility traces, etc.

- Can be used to extract “typical” behaviour

  - *Advantage*: true behaviour, high level of detail, can be used to validate simulation model

  - *Disadvantage*: small amount of data, generalization difficult

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Empirical Distributions

- Generalization of recorded data
  - Assumption: samples follow the same distribution function
  - Let $x_1...x_n$ be a set of characteristic values with their frequencies $h_i$ taken from a sample
  - Empirical distribution function (ECDF) for the sample:

$$F(t) := \begin{cases} 0, & \text{if } t < x_1 \\ \sum_{j=1}^{i} h_j, & \text{if } x_i \leq t < x_{i+1}, \ i \in \{1, \ldots, k - 1\} \\ 1, & \text{if } x_k \leq t \end{cases}$$
Empirical Distributions

- Example:

Deaths by horsekick in Prussian cavalry corps, 1875-94

Cumulative density

deathsPerYear
Empirical Distributions

- Sometimes not all characteristic values can be observed ... but it is known how many observations fall into a certain interval
- Data can be grouped into intervals
- Distribution function can be defined as follows:
  - interval borders define points of the function
  - interpolate between the points

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Analytical distributions (1)

- Empirical distributions can be approximated by analytical distributions
Analytical distributions (2)

- How to choose a distribution function that matches the empirical distribution?
  - Similarity of the diagrams (ECDF plot and analytical distribution function)
  - Knowledge about the process or the type of events (e.g. independent arrivals are modeled best by a Poisson process)
  - Quality criterion: Goodness of fit test
Probability Distributions Revisited (1)

- Let X be a random variable

- Cumulative distribution function (CDF):
  - $F(x) = \text{Prob}[X \leq x]$
  - Range of values: [0,1]
  - Monotonically increasing

- Probability density function (PDF):
  - $f(x) = \text{Prob}[X = x]$
Probability Distributions Revisited (2)

- Example: Normal distribution

\[
F(x) = \int_{t=0}^{x} f(t)
\]
Probability Distributions Revisited (3)

- Example: Poisson distribution with $\lambda=3$

$$F(x) = \sum_{t=0}^{x} f(t)$$
# A Few Discrete Distributions

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>[a,b]</td>
<td>All outcomes in the interval [a,b] have the same probability</td>
</tr>
<tr>
<td>Bernoulli</td>
<td>p</td>
<td>“coin flip” with success probability $p$</td>
</tr>
<tr>
<td>Binomial</td>
<td>n, p</td>
<td>Number of successes of $n$ independent Bernoulli trials with success probability $p$</td>
</tr>
<tr>
<td>Geometric</td>
<td>p</td>
<td>Number of successive failures of independent Bernoulli trials until the first success</td>
</tr>
<tr>
<td>Poisson</td>
<td>$\lambda$</td>
<td>Number of events within a unit time interval, if the time between event arrivals is exponentially distributed with rate $\lambda$.</td>
</tr>
</tbody>
</table>
# A Few Continuous Distributions

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>[a, b]</td>
<td>All outcomes in the interval [a, b] have the same probability (usually the interval is [0, 1])</td>
</tr>
<tr>
<td>Exponential</td>
<td>λ</td>
<td>Time until the next arrival of an event, if events arrive independently at a rate of λ events per unit time (cf. Poisson)</td>
</tr>
<tr>
<td>Normal</td>
<td>μ, σ</td>
<td>The mean of independent repetitions of <em>any</em> random experiment converges towards the normal distribution (mean μ, standard deviation σ).</td>
</tr>
<tr>
<td>Pareto</td>
<td>α</td>
<td>Can be used to describe the time between event occurrences (cf. Exponential)</td>
</tr>
</tbody>
</table>
Goodness of Fit

- How well do two distributions match?
  - Compare distributions graphically
  - Goodness-of-fit tests
    - $\chi^2$ test (chi square test)
    - Kolmogorov-Smirnov test
    - ...
  - cf. Exercise 8
Chi Square Test (1)

- Comparison of the empirical histogram with a conjectured distribution
- Null Hypothesis: Sample data follows a certain distribution
- Method:
  - Divide the observations into k cells
  - Calculate the frequency $f_i$ of observations in each cell
  - Calculate the expected frequency $n \cdot p_i$ according to the conjectured distribution
  - Calculate chi square: $\chi^2 = \sum_{i=1}^{k} \frac{(f_i - np_i)^2}{np_i}$
  - Reject the hypothesis, if $\chi^2$ is too large

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Chi Square Test (2)

- The test statistics is assumed to be chi-square distributed with \( k-1 \) degrees of freedom.
- For a significance level \( \alpha \), the hypothesis should be rejected if \( \chi^2 > (1-\alpha)\)-quantile of the \( \chi^2 \) distribution with \( k-1 \) degrees of freedom.
- \( \chi^2_{k-1,1-\alpha} \) can be looked up from a table
- Statistics programs calculate the significance (p-values)
- Rule of thumb for choosing intervals:
  at least 3 intervals; \( n \cdot p_i \geq 5 \) for most of the intervals

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Quantile of the $\chi^2$ Distribution

Example:

$\chi^2$ distribution with 3 degrees of freedom (df=3)

1-$\alpha$ Quantile:
$F_{\chi^2}(x) = 1-\alpha$

area = 0.95

1-$\alpha$ quantile for $\alpha = 0.05$ (significance level)

$x = 5.99$
The $\chi^2$ Distribution

The graph shows the probability density function (PDF) of the $\chi^2$ distribution for different degrees of freedom (df). The curves are labeled with their respective degrees of freedom: df=1, df=2, df=3, and df=4. The x-axis represents the variable $x$, and the y-axis represents the probability density $f(x)$. The graph illustrates how the shape of the $\chi^2$ distribution changes with the degree of freedom.
P-values

- P-value describes the significance of a test result
  - Probability of getting a result that is at least that extreme (all under the assumption of a null hypothesis)
  - The lower the P-value the less likely the result

**Example**: Chi square test for a coin flip.
- Null hypothesis: the coin is fair
- 20 heads, 20 tails: $\chi^2 = 0$, p-value = 1
- 25 heads, 15 tails: $\chi^2 = 2.5$, p-value = 0.1138
- 30 heads, 10 tails: $\chi^2 = 10$, p-value = 0.001565
Kolmogorov-Smirnov Test (1)

- Comparison of two distributions
  - sample distribution and analytical distribution (one-sample K-S test)
  - two sample distributions (two-sample K-S test)

- Calculates the maximum distance between empirical and conjectured distribution

- Reject hypothesis, if $d_{\text{max}} > d_\alpha$
  ($d_\alpha$ can be looked up from a table)

- Mainly used for continuous distributions
Kolmogorov-Smirnov Test (2)

- Result of the K-S test: Maximum difference in value

![Diagram showing the Kolmogorov-Smirnov test with the maximum difference (d_max) highlighted between the empirical and Poisson distribution curves.]
Models for Arrival Processes

- At what time does a request arrive in the system?
  - customers, packets, ...

- Arrival Process = stochastic description of arrivals

- Some types of arrival processes
  - Constant bit rate (CBR)
  - Renewal process
  - Markov process

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Constant Bit Rate

- **CBR**: Simple model for regular events

- **Characteristics:**
  - Generate load at fixed time intervals
  - Workload per task can be varied

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Renewal Processes

- When interarrival times are independent and identically distributed random variables

- **Special cases:**
  - Poisson process: inter-arrival times are exponentially distributed (continuous time)
  - Bernouilli process: inter-arrival times are geometrically distributed (discrete time)

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Poisson Process (1)

- Discrete number of events, continuous time
- Event occurrences are independent
- Number of events in a time interval of length $t$ follows a Poisson distribution:

\[
\Pr_{\lambda,t}[X = k] = \frac{\lambda^k}{k!} e^{-\lambda t}
\]

- $\lambda t$ = expected number of occurrences in time interval $[0,t]$
Poisson Process (2)

- Time between event occurrences (inter-arrival time) is \textit{exponentially distributed}:
- Let $X$ be a Poisson random variable with parameters $\lambda$ and $t$, and $A$ a random variable that describes the time until the next arrival. An inter-arrival time $A > t$ is equivalent to $X = 0$ for the interval $[0,t]$.
  - $\Pr[A>t] = \Pr[X=0] = e^{-\lambda t}$
  - $\Pr[A\leq t] = 1 - e^{-\lambda t} =: F(t)$ (cumulative dist. function of $A$)
  - $f(t) = F'(t) = \lambda e^{-\lambda t}$ (probability density function)
- $A$ follows an exponential distribution
Markov Processes (1)

- Similar to renewal processes, but with history, i.e. there are interdependencies between inter-arrival times
  - State-based
  - Probability of state transitions (defined in a prob. matrix)
  - Transitions depend *only* on the current state.
  - Continuous time: A Markov process remains in a state i and holds this state for a random exponentially distributed time. Then it switches to state j with probability $p_{ij}$.
  - Discrete time: State transition after each time step *(Markov chain)*

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Markov Processes, Example (1)

- **Example**: Modeling gaps and bursts of a video stream (load model)

```
\[ p_{GB} = \text{probability to switch to the burst state} \]
\[ p_{BG} = \text{probability to switch to the gap state} \]
```

**Gap**: no packet injection or low arrival rate process

**Burst**: CBR or high arrival rate process
Markov Processes, Example (2)

- Gilbert-Elliott Model for bit errors (fault model):
  - good and bad state of the channel
  - Bit errors occur in the bad state

![State Transition Diagram]

1 - \( p_{\text{better}} \)  \[ \text{good} \] \( p_{\text{worse}} \) \[ \text{bad} \] 1 - \( p_{\text{worse}} \)

\[ p_{\text{better}} \]

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Gilbert-Elliott, Implementation

- Not efficient: simulating bit errors by applying the Gilbert-Elliott model bit by bit.
- Better: Determine the number of bits until the next state changes. This number is geometrically distributed.
- Even better: If the position of bit errors is not important, only the number, then it is more efficient to determine the number of bit errors in a certain time interval.

[H. Karl, Leistungsbewertung und Simulation, Uni Paderborn, 2007]
Lessons learned

- Simulation results are only an *estimation* of the true behaviour

- Pitfalls: bad random number generators, bad seed selection, insufficient simulation time

- Be careful when using aggregated information and interpreting results