Ad Hoc Networking

Models and Methods

Holger Hermanns
Sven Johr

Dependable Systems & Software
Universität des Saarlandes

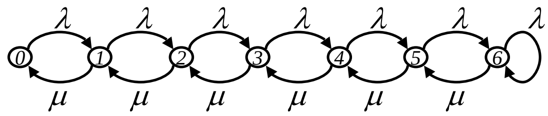
WS 03/04
Lecture 6: Analysis of Stochastic Processes

At a small Mc Donalds

- Customers arrive at a certain frequency, say approximately 1 customer per five minutes.

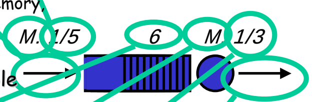
  arrival rate λ = 1/5 min
- Service requires, say, three minutes.

  service rate μ = 1/3 min
- At most six customers can wait inside the Mc Donalds (it's a small one).



Mc Donalds as a queueing system

- arrival stream: hungry people
  - 1/5 customers per minute,
  - Markovian, i.e. no memory,
- queue: space available
  - queue capactiy
- service station: the guy who sells hamburgers
  - 1/3 hamburgers per minute,
  - Markovian, i.e. no memory,
- departure stream: stuffed people



M/M/1/6/∞/FIFO

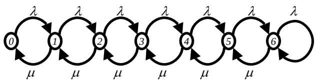
Analysing stochastic models

What would you like to know?

- **Or**, what is your measure of performance?
  - mean number of customers waiting in the Mc Donalds?
  - mean time a customer has to wait?
  - percentage of time the Mc Donalds is utilised by someone?
  - number of customers served per minute?
  - percentage of customers that are lost, due to lack of space?
  - profit made?
  - number of wealthy customers lost?
  - ...

Standard performance measures

- mean queue length,
- mean waiting time,
- throughput,
- probability of loss,
- utilization,
- ...



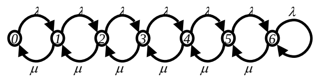
... and dependability measures

- availability,
- point reliability,
- performability,
- ...

Calculating performance measures

- all these performance (and dependability) measures can be computed on the basis of
  - transient state probabilities P(X(t) = s),
  - steady-state probabilities $\lim_{t \to \infty}$ P(X(t)=s),

  and possibly state/transition labels.



- Computation of ***state probabilities*** is the main technical issue.

- Recall: state probabilities describe the likelihood of being in a certain state (at a certain time instant)

Calculating state probabilities

- There are three fundamentally different ways to calculate these state probabilities:
  - *analytical* solution,
  - *numerical* solution,
  - *discrete event simulation*.

Analytical solution

- express the state probabilities (or even measures directly) as *closed formulae* in the parameters of the model

example: utilization of the Mc Donalds $\rho(\lambda,\mu) = \lambda / \mu$

provided that $\lambda < \mu$, and that the queue length may become larger than 6, namely infinite

- *positive:*
  - very accurate
  - very fast, and simple
- *negative:*
  - only for highly restricted classes of stochastic processes
  - requires study of scientific literature, to find specific formulae

Numerical solution

- state probabilities are obtained by an exact or approximative algorithm where model parameters are instantiated with numerical values.

example: state probabilities of the Mc Donalds are obtained by (e.g.) Gauss elimination of a 7×7 matrix based on 1/3 and 1/5 entries.

- *positive:*
  - accurate, up to numerical precision
- *negative:*
  - only reasonable for *finite Markov chains*
  - number of states is a limiting factor (about 10^8)

Discrete Event Simulation

- the stochastic model is mimicked by a simulator rolling dices and producing statistics of *simulation time* spent in states. The fraction of *simulation time* spent in a particular state is used as an estimate for the state probability.

example: Let a lot of virtual people enter a virtual Mc Donalds and let them ask for virtual hamburgers. Do this 100000 times faster than real time, or better, as fas as you can. Compute the fraction of time when there is someone in the resto.

- *positive:*
  - very general, suitable for arbitrary stochastic models
- *negative:*
  - good accuracy usually requires long (or very long) simulation runs: accuracy grows with the square root of the number of runs.

Rules of thumb

- *Analytical solution* allows very *quick* and very *precise* insight in your model, but the model tends to be a very *loose approximation* of reality.

- *Simulation* allows relatively *slow*, *rough* and *costly* insight in a *single* instance of your model, but the model can have a *close correspondence* to reality.

- *Numerical solution* allows *quick* and *precise* insight in a *single* instance of your *Markov chain* model, which usually is an *approximation* of reality (due to absence of memory).

Stochastic modelling and analysis

The standard procedure:

- construct a model,
- determine your performance measure of interest,
- choose a solution method:
  - analytical,
  - numerical, or
  - simulation,
- fix model parameters ($\lambda,\mu,\ldots$),
- derive performance measure.

play with the model

Why play with model parameters?

- to pose "what if" questions
  - *perturbation analysis*
- to see how performance changes if parameters change
  - *sensitivity analysis*
- to find the best performance (tuning)
  - *optimisation*

Rules of thumb revisited

- *Analytical solution* allows very *quick* and very *precise* insight in your model, but the model tends to be a very *loose approximation* of reality.
- *Simulation* allows relatively *slow*, *rough* and *costly* insight in a *single* instance of your model, but the model can have a *close correspondence* to reality.
- *Numerical solution* allows *quick* and *precise* insight in a *single* instance of your *Markov chain* model, which usually is an *approximation* of reality (due to absence of memory).

In order to *optimise* (etc.) that computation has to be repeated many times

Discrete Event Simulation

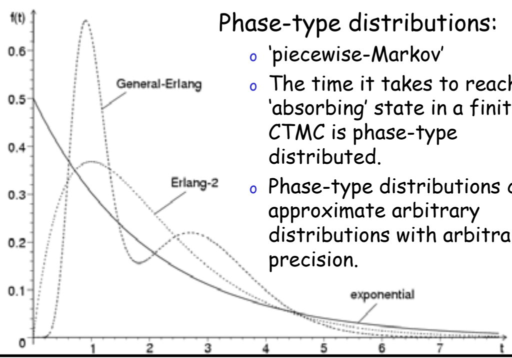
- the stochastic model is mimicked by a simulator rolling dices and producing statistics of *simulation time* spent in states. The fraction of *simulation time* spent in a particular state is used as an estimate for the state probability.

example: Let a lot of (virtual) people enter a virtual Mc Donalds and let them ask for a virtual hamburger. Do this 100000 times faster than real time, or better, as fas as you can. Compute the fraction of time when there is someone in the resto.

What else is there beyond Markov Chains

- *pro's:*
  - very general, suitable for arbitrary stochastic models
- *con's:*
  - good accuracy usually requires long (or very long) simulation runs: accuracy grows with the square root of the number of runs.

Virtually beyond the Markov property



Phase-type distributions:

- 'piecewise-Markov'
- The time it takes to reach an 'absorbing' state in a finite CTMC is phase-type distributed.
- Phase-type distributions can approximate arbitrary distributions with arbitrary precision.

Beyond the Markov property: Semi-Markov Chains

- Markov chain on a set of states {0,1,…}, that whenever entering state i
  - the next state that will be entered is j with probability P_{ij}
  - given that the next state entered will be j, the time it spends at state i until the transition occurs is a RV with distribution F_{ij}

Beyond the Markov property: Semi-Markov Chains

- Markov chain on a set of states {0,1,…}, that whenever entering state i
  - the next state that will be entered is j with probability P_{ij}
  - given that the next state entered will be j, the time it spends at state i until the transition occurs is a RV with distribution F_{ij}
- $\{Z(t): t \geq 0\}$ describing the state the chain is in at time t: *Semi-Markov Chain*

➧ future depends on present state *and time spent* in the state

➧ memory is lost on state change

Semi-Markov Chains

- *Holding time*: time spent at state i, before making a transition
- Probability distribution function of *holding time:*

$$H_i(t) = P\{T_i \leq t\} = \sum_{j=0}^{\infty} P\{T_i \leq t \mid \text{next state } j\} P_{ij} = \sum_{j=0}^{\infty} F_{ij}(t) P_{ij}$$

$$E[T_i] = \int_0^{\infty} t \, dH_i(t)$$

Semi-Markov Chains

- *Holding time*: time spent at state i, before making a transition
- Probability distribution function of *holding time:*

$$H_i(t) = P\{T_i \leq t\} = \sum_{j=0}^{\infty} P\{T_i \leq t \mid \text{next state } j\} P_{ij} = \sum_{j=0}^{\infty} F_{ij}(t) P_{ij}$$

$$E[T_i] = \int_0^{\infty} t \, dH_i(t)$$

- *Let* X_n describe the n^{th} state visited. $\{X_n: n=0,1,\ldots\}$
  - is a DT Markov chain: embedded Markov chain
  - has transition probabilities P_{ij}

Semi-Markov Chains

- *Holding time*: time spent at state i, before making a transition
- Probability distribution function of *holding time:*

$$H_i(t) = P\{T_i \le t\} = \sum_{j=0}^{\infty} P\{T_i \le t \mid \text{next state } j\} P_{ij} = \sum_{j=0}^{\infty} F_{ij}(t) P_{ij}$$

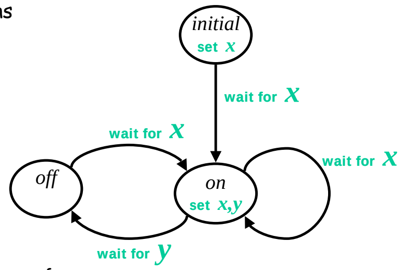
$$E[T_i] = \int_0^{\infty} t \, dH_i(t)$$

- *Let* X_n describe the n^{th} state visited. $\{X_n: n=0,1,\ldots\}$
  - is a DT Markov chain: embedded Markov chain
  - has transition probabilities P_{ij}
- $\{Z(t): t \ge 0\}$ is completely determined by

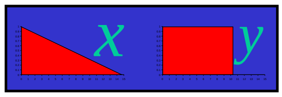
P_{ij}, F_{ij} and $Z(0)$

Beyond Semi-Markov: Stochastic models *with* memory

- states, transitions
- labels
  - of states
  - of transitions

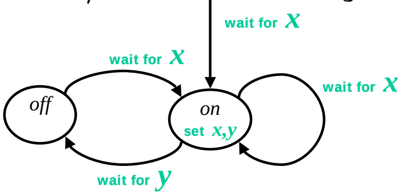


- *clocks*
  - serve as the memory of time orthogonal to states.
  - are sampled from distributions, count down.

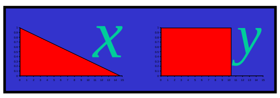


A light in the stairway

- The light is turned *on* if someone enters the stairway.



- It goes *off* after exactly 10.3 minutes.
- People arrive randomly, at least every 15 minutes, with equal probability for each time instant.



Underlying Model: Automaton

Automaton: (E, X, Γ, f, x_0)

E : Event Set

X : State Space

$\Gamma(x)$: Set of *feasible* or *enabled* events at state x

f : State Transition Function $f: X \times E \to X$

(partial, undefined for events $e \notin \Gamma(x)$)

x_0 : Initial State, $x_0 \in X$

$$\{e_1, e_2, \ldots\} \longrightarrow \boxed{f(x,e) = x'} \longrightarrow \{x_1, x_2, \ldots\}$$

Timed Automaton

Add a ***Clock Structure V*** to the Automaton: $(E, X, \Gamma, f, x_0, \mathbf{V})$

where: $\mathbf{V} = \{\mathbf{v}_i : i \in E\}$

and $\mathbf{v}_i$ is a ***Clock or Lifetime*** Sequence: $\mathbf{v}_i = \{v_{i1}, v_{i2}, \ldots\}$
one for each event i

$$\{\mathbf{v}_1, \mathbf{v}_2, \ldots\} \longrightarrow \boxed{f(x, e') = x'} \longrightarrow \{x_1, x_2, \ldots\}$$

NEXT EVENT

Need an *internal mechanism* to determine *NEXT EVENT e'* and hence *NEXT STATE* $x' = f(x, e')$

How the timed automaton works

current state

$x \in X$ with feasible event set: $\Gamma(x)$

current event

e that caused transition into x

current event time

t associated with e

Associate a *clock value (or:* residual lifetime) y_i with each feasible event $i \in \Gamma(x)$

How the timed automaton works

NEXT/TRIGGERING EVENT *e'* :

$$e' = \arg\min_{i \in \Gamma(x)} \{y_i\}$$

NEXT EVENT TIME *t'* :

$$t' = t + y^*$$

where: $y^* = \min_{i \in \Gamma(x)} \{y_i\}$

NEXT STATE *x'* :

$$x' = f(x, e')$$

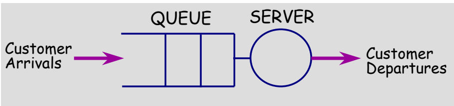
How the timed automaton works

Detemine new *clock values* y_i'
for every event $i \in \Gamma(x)$

$$y_i' = \begin{cases} y_i - y^* & i \in \Gamma(x'), i \in \Gamma(x), i \neq e' \\ v_{ij} & i \in \Gamma(x') - \{\Gamma(x) - e'\} \\ 0 & otherwise \end{cases}$$

where: v_{ij} = new lifetime for event i

Timed automaton example



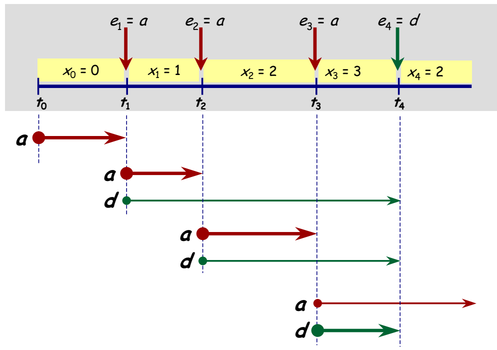
$E = \{a, d\}, \quad X = \{0, 1, 2, \ldots\}$

$\Gamma(x) = \{a, d\}, \ \text{for all } x > 0$
$\Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & e' = a \\ x - 1 & e' = d, \ x > 0 \end{cases}$$

Given input : $\mathbf{v}_a = \{v_{a1}, v_{a2}, \ldots\}, \ \mathbf{v}_d = \{v_{d1}, v_{d2}, \ldots\}$

Timed automaton sample path



Stochastic timed automaton

- Same idea with the Clock Structure consisting of ***Stochastic Processes***
- Associate with each event i a ***Lifetime Distribution*** based on which $\mathbf{v}_i$ is generated

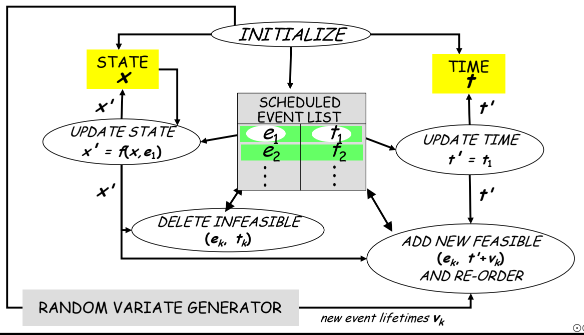
$\mathbf{v}_i = \{v_{i1}, v_{i2}, \ldots\}$

Generalized Semi-Markov Process (GSMP)

In a simulator, $\mathbf{v}_i$ is generated through a pseudorandom number generator

Discrete Event Simulation (DES)

…is simply a computer-based implementation of the sample path generation mechanism described so far



DES Simulation Classification

Terminating Simulations:

- stop when given state is reached
- stop when given event count is reached
- stop when given time is reached

Example: UdS VoIP performance between 9-10AM daily

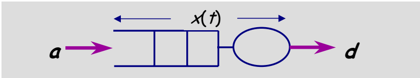
Non-terminating Simulations:

- System is perpetually operating
- We are interested in steady-state performance

Example: Average performance of UdS VoIP

DES Performancs metrics

1. Event Counts:
   - number of events in a given time interval
   - number of events in a random time interval
   - number of events satisfying a given condition



- $N_d(T)$ = no. departures in fixed T
- ***THROUGHPUT*** = $N_d/T(N_d)$ (N_d fixed, $T(N_d)$ random)
- ***QUALITY-OF-SERVICE***: $N_a(K,T)$ = no. arrivals that see $x > K$ in fixed T

DES Performance metrics (more)

2. Event times:
   - event time for a given event count
   - interval between specified events

- $T(N_d)$ = time required to obtain N_d departures (N_d fixed)
- ***Response time*** or ***Delay*** $D = T(N_d) - T(N_a)$ ($N_d = N_a = N$ fixed)

DES Performance metric estimation

In the GSMP setting, performance metrics are parameters of stochastic processes:

Expectations:

Expected Response Time or *Expected Delay* :

$E[D] = E[T(N_d) - T(N_a)]$ ($N_d = N_a = N$ fixed)

Probabilities:

$P[D > t] = E[1(D > t)]$ (measures Quality-of-Service)

INDICATOR FUNCTION

DES Performance metric estimation

- Terminating Simulations:
  - Expectations estimated over multiple sample paths (ensemble averaging)
- Non-terminating Simulations:
  - Expectations estimated over long time periods (time averaging)

Discrete Event Simulation: Summary

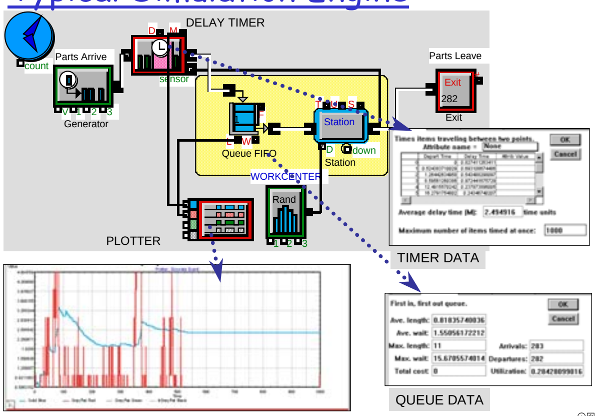
- A systematic way to construct sample paths.
- Simulators are normally equipped with data collection and output processing capabilities

  ☞ used to estimate performance metrics of stochastic models of virtually arbitrary complexity

Limitations:

- Slow and costly, limited real-time capabilities
- Requires expertise to interpret statistical results

Typical Simulation Engine



Assignements

1. Can you rephrase a CTMC as an SMC? If so, what is the embedded DTMC of this SMC?
2. What happens if in a GSMP, two enabled events have identical residual lifetimes? Can/must this be avoided?