Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp458-466)

Stochastic Network Simulation for Reliable Results

TAPIO FRANTTI
Telecommunication
VTT Technical Research Centre of Finland
Kaitoväylä 1, FIN-90571, Oulu
FINLAND
i

Abstract: - During the last decades stochastic simulation methods have been utilized more and more in different research fields to describe complicated real-world phenomena. Reasons for the popularity are dramatic increase in processing power and significant decrease of price of computing systems. However, the main reason may be that the probability theory is a well-known tool for presentation and processing of stochastic information. In this article are described fundamental stochastic features of communication network simulation models. Especially, a concept of convergence time is considered and its numerical evaluation is defined by an example simulations. Network models considered here are event based discrete time models, which utilise Markov chain theory for state transitions and Monte Carlo method for duration times of events. Numerous reported case studies of stochastic simulations do not contain any information about a convergence time of the model. Therefore, it is possible that the reported results are randomly biased due to too short simulation (shorter than the convergence time) periods as illustrated in this article by examples.

Key-Words: - communication network, stochastic simulation, convergence time, ad hoc, random number.

1 Introduction

Simulation is a widely used method in the research and planning of communication networks. It is essentially a modeling tool, which can yield immense benefits to the researchers or designers of the system. Simulation lets them to draw appropriate conclusions and to do right decisions before major investments are made. The validity of the obtained conclusions depends greatly on the accuracy of the model and the correct use of it as well as the correct interpretation of the achieved results. Accuracy of the model refers to how strictly it captures the actual operations of the real system. Correct use of the model and interpretation of the results means that fundamental limits of the model are understood and taken into consideration.

The purpose of simulation models is to evaluate real processes numerically usually by a computer program. For computer simulation it is necessary to do a set of assumptions in the form of mathematical and logical relationships. The assumptions constitute a model, which can be used to achieve understanding from the system behavior.

Simulation models can be classified several ways, like *static* vs. *dynamic*, *deterministic* vs. *stochastic* and *continuous* vs. *discrete* models. In static simulation models time does not play any role whereas dynamic models evolve over time. Deterministic models do not contain any random/probabilistic components meanwhile stochastic models are based on probability distributions. In continuous models state variables change continuously over time whereas in discrete models systems evolve over time at separate points in time.

Usually simulation models are developed either for system design, system management or training. System design includes development of the new system and improvement of an existing one. System management models are used for scheduling and control of real-time systems. Training refers to task and operation training of systems.

Instead of simulations, in some cases communication networks can also be modeled, analysed and designed using either *measurements* or *mathematical analysis*. Measurements offer direct means for performance evaluation. They are also the most expensive way due to need of a new network. Experimentations with an existing network may also be economically very unfeasible if they suspend the operations of commercial network during the tests. Mathematical analysis necessitate a high degree of abstraction and requires considerable amount of skill. Analytic models can usually be solved quickly and they can be effective when they are carefully applied observing their limitations.

In this article is described fundamental features of simulation models. Especially, the concept of convergence time is considered. Practical model-based evaluation of it is defined by example

simulations. Network models considered here are event based discrete time models, which utilise Markov chain theory for state transitions and Monte Carlo method for duration time definitions of events.

2 Discrete Event Simulation

Discrete event stochastic simulation is commonly used for the modelling of communication networks and protocols. It offers flexibility of performance modelling at any level of detail [5]. Network components like communication links, repeaters, routers and transceivers are presented using computer program modules. Events of the networks, like transmission, arrival, routing, loss, rejection and delays of packets and link failures are mimicked in the simulation program execution.

Despite of wide range of applications, discrete event stochastic simulation models share a number of common features, like system clock, event lists, statistical counters and initialization, timing, as well as event, library, random sampling and report routines. Here we consider especially *simulation clock* and *random sampling* of stochastic events because of their definitive effects for convergence time, reliability and plausability of the model.

2.1 Simulation clock

Simulation clock is a variable, which gives the current synchronised value of simulation time for network elements. It is required to maintain simulation time information in dynamic models.

Simulation models have two fundamentals mechanisms to advance time: *next-event time* and *fixed increment time* advance. The next-event time approach is used in the most simulation cases instead of the fixed increment approach because the most real system events does not occur at fixed time intervals. In the beginning of simulation, the simulation clock is initialized to zero and the occurrence time of the next event is determined. The simulation clock is then advanced to the time of next event and the state of the system is updated accordingly. Then the time is advanced with the duration time of the event or the latest event if another event is occuring at the same time (parallell events). This is continued until predefined stopping condition is fulfilled.

2.2 Random sampling

Communication networks as large-scale systems are very complex. A simulation of them is, in an abstract level, parallel series of deterministic and stochastic events with predefined or random occurrence and duration times. Stochastic duration times are drawn from specified distributions using random number generator in which numbers are usually uniformly distributed on the interval [0,1] and do not exhibit any correlation with each other. The distributions can be some known standard distributions like normal and uniform distributions or nonstandard experimental distributions. A given random number indicates the duration time of the event. It is defined from the cumulative distribution of the duration time density distribution. In the Figure 1 a given random number, *i.e.*, probability is 0.395 and it indicates to a specific point in a cumulative distribution of an example event's duration time. The respective duration time is got from the horizontal axe, 9.0 milliseconds. This is called Monte-Carlo technique.

Deterministic duration time of the event refers to the fixed duration time. The occurrence time of an deterministic event is either triggered by another event(s) or transition to a specific state whereas stochastic event has probabilistic nature (like transfer error has) and it can be triggered also by probability calculation.

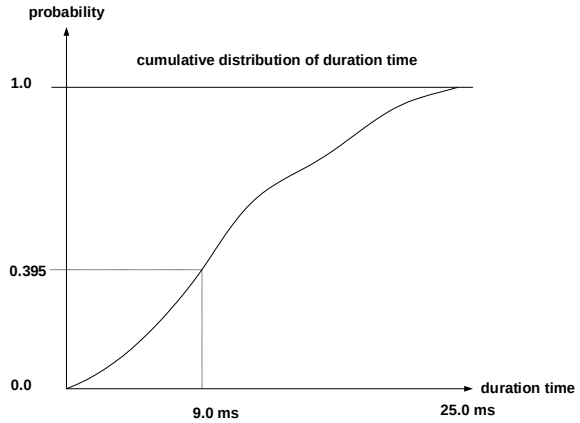

Fig.1. A given random number indicates duration time of the event.

3 Random Numbers

The word *random* is usually reserved for the output of an intrinsically random physical processes, but in this paper it is used to describe *pseudorandom* numbers generated by computer programs. A definition of *randomness* in the context of computer-generated sequences can be stated so that the deterministic program that produces a random sequence should be different from and in all measureable respects statistically uncorrelated with the computer program that uses its output. Hence,

two different random number generators ought to produce statistically the same results when utilized in a particular application program [11]. There exist a certain list of statistical tests for testing any correlations that are likely to be detected by an application program. As a reference to the topic, see Knuth [6] and Bratley [3].

A reliable source of random *uniform deviates* is an essential building block for any sort of stochastic modeling or Monte Carlo computer work [11]. Uniform deviates are random numbers that lie within a specified range (typically from 0 to 1). However, there is other sorts of random numbers, *e.g.*, numbers drawn from a normal distribution with a specified mean and standard deviation.

3.1 Random number generation

Each number in a computer memory consist of *n* bits. In a signed on-off coding, for example, one bit is used for the *sign* and n-1 bits for the *magnitude*. Therefore, for example for the integers, the largest number is 2^{n-1}-1.

System-supplied random number generators are almost always *linear congruential generators*. They generate a sequence of integers I_1, I_2,... each between *0* and *m-1* by the recurrence relation [11]:

$$I_{j+1} = aI_j + c \ (mod\ m), \qquad (1)$$

where *m* is modulus, *a* and *c* are positive integers called the multiplier and the increment, respectively. The recurrence will eventually repeat itself with a period no greater than *m*.

Implementation of a simple congruential generator, *e.g.*, in many ANSI C libraries as *rand ()* function is quite flawed because ANSI C does not specify a standard algorithm for *rand ()* function. It only states than an implementation of the *rand ()* should generate the best possible random sequence. However, ANSI C committee has published an example algorithm represented in [11]:

```
unsigned long next=1;
int rand(void) {
next = next  x 1103515245 + 12345;
return (unsigned int) (next/65536)
}
void srand (unsigned int seed) {
next = seed;
}
```

This corresponds to the equation (1) above. ANSI C committee also stated that "try to improve the published algorithm". Unfortunately some widespread modifications in fact ruins the generator instead of improving it.

This kind of linear congruental method is very fast requiring only a few operations per function call. Hence, it is in widespread use under various names. However, it is not free from the sequential correlation on successive calls [2][11].

Sequential correlation can be visualized if *k* random numbers at a time are used to plot points in a *k* dimensional space. Then the points will not tend to fill up the *k* dimensional space but rather lie on *k-1* dimensional planes (at most $m^{1/k}$ planes; if constants *m*, *a*, and *c* are not carefully chosen there will be fewer than that). For example, if *m* is 32786 (2^{16}), the number of planes on which triple of points lie in three-dimensional space is about the $32786^{1/3}$ or 32. Even if *m* is close the largest representable integer in 32 bit computer, the number of planes is only about 1600 [11].

Correlation is not the only weakness of this kind of generators but such generators have also the least significant bits much less random than the most significant bits. Therefore, *e.g.*, for random integer between the 1 and 10 should always be generated using high order bits, like in ANSI C:

```
j=1+(int) (10.0 x rand()/(RAND\_MAX+1.0));
instead of
j=1+(rand()%10); .
```

In [9] have been analysed a large number of random number generators. They present, *e.g.*, *minimal standard random number generator*, which is satisfactory for the majority of applications. Even if, this algorithm has subtle serial correlations present it can be removed by shuffling the output so that a random deviate derived from the j^{th} value in the sequence, I_j, is the output on a randomized later call instead of the j^{th} call.

3.2 Example random number generator

In this subsection is presented an improvement for the above presented linear congruental algorithm. It is shown in the literature that the simple multiplicative congruential algorithm can be as good as any of the more general linear congruental algorithm that have $c \neq 0$ in the equation (1) above. Therefore, in a computer software running in a 32 bit computer, random integer number I_1 between 0 and 2^{31}-1 can be produced from an initial integer I_0 by multiplying I_0 with a suitable number. In [9] it is proposed values 48271 for *a* and 2^{31}-1 for *m*. This in general produces an integer which distinctly exceeds the maximum value for a 32 bit integer. The product

is put back to the interval [0, 2^{31}-1] by a modulo operation [2]:

$$I_1 = 48271 \times I_0 \text{ modulo } 2^{31}-1 \quad (2)$$

For the computer, one has to ensure that the result is not negative because some of them simply omit the leading bits of the product. Hence, for the 32 bit computer, the following pseudocode statements produce from a random integer (stored as variable *I*) another integer (stored at the end again under the name *I*) and a real number *X* between zero and unity:

I = I x 48271, I is a random integer
if (I < 0) then I = 2^{31}-1
X = I x $(2^{31}-1)^{-1}$

By repeating this process again and again we get a series of random numbers between zero and one, which are approximately random and approximately homogenous. However, the resulting series is reproduciple, if we start with the same initial number (seed). It also repeats itself after a period of at most 2^{31}-2. The initial value must also be a positive odd integer. Therefore, *I* should be

I = 2 x |I| + 1

after *I* is set initially. If it is not a very large, one needs several iterations to get a series of numbers *X* which are not very small. In order to avoid that one can use a short warm-up loop which performs several iterations. One choice for the random initial value (seed) *I* is a thousand part of second on computer's internal clock scaled appropriately to get it large enough and to changed to odd integer. Using random seed also decrease the possibility of synchronization (parallel units do their sampling procedure identically, *i.e.*, sampling probabilities correlate) in parallel computing.

However, the consecutive random numbers have strong correlations between them. This problem should be solved, *e.g.*, by mixing or shuffling the outputs of two random number generators. This can be done, for example, by determining randomly an integer *K* between [0,255] and extracting a random number from a table of 256 random numbers and replacing it with a freshly calculated number from the above described random number generator:

L = random_number_table[0]
I = I x 48271, I is a random integer
if (I<0) then I = 2^{31}-1
J = L/(1+(2^{31}-1)/32)
L = random_number_table[J]
if (L<0) then L = 2^{31}-1
random_number_table[J] = I x $(2^{31}-1)^{-1}$

Usually computers with 60 bits per word in the above described mixed random numbers seem to be good enough for all practical experiments in statistical communication network simulations [2].

4 Convergence Time

In quantitative simulation studies attention should be put to accuracy of the final results. However, it is surprising to notice that in only a few reported simulation studies statistical analyses are carefully performed to control the statistical errors of their final results. In [4] has pointed out already in 1990 that no any other field of science and engineering has not taken such a liberty with empirical data. [8] also points out that simulation without careful statistical analysis of output data can provide erroneous results. This alarming trend has continued despite of the early warnings.

Random sampling of the series of stochastic distributions, standard or experimental, leads to an output value which depends on the number, types and parameters of the distributions. The output value approaches or converges a constant value as a function of simulation time. Deterministic phases increase the total expected value but do not change variance or convergence time. However, series of deterministic events, such as predefined tracks of movements of mobile transmitters in ad hoc networks, may act as a stochastic distribution.

The convergence time refers to the minimum simulation time of the model so that by simulating equal or longer period the reference value achieved with fixed set of simulation parameters do not oscillate significantly. According to the Central Limit Theorem (consult, *e.g.*, [12] for a general view or [1] for more thorough mathematical explanation) a series of probability distributions obey normal distribution as a whole. The convergence time refers to the expected value of this distribution. Therefore, the value of reference variable in a specific simulation model converge to this value.

In a simulation model development process it is mandatory to define the convergence time before simulation runs to get reliable results. The influence of some parameter to the reference value can not be judged with shorter simulation times. This is due to the fact that reference value oscillates according to the random values of events probabilities and duration times evaluated in different sequential and

parallel phases. If the simulation time exceeds the convergence time, the effect of the parameter under study can be seen on the output of the reference value. In a case that parameter has influence to the simulated system's behavior, the expected value and/or variance of the reference variable is/are changed as illustrated in section *Results*.

5 Internet Traffic Simulation

In this section, is presented developed theoretical Internet data traffic model. In the literature ([7], [10] and [14]) it has been shown that LAN (Local Area Network) traffic is statistically *self-similar* (process in larger scale is a copy of itself in smaller scales) and that none of the commonly used traffic models are able to model this fractal-like behavior. Moreover, in [7] it has also been claimed that such a behavior has serious implications for the design, control, and analysis of high-speed networks, and that aggregating streams of such traffic typically intensifies the self-similarity ("burstiness") instead of smoothing it. Therefore, the proper simulation model uses *Pareto distributed traffic* for the evaluation of the developed solutions. Self-similar traffic can be generated by multiplexing several Pareto distributed packet trains and silence periods between packet trains. The Pareto distributed traffic model has Pareto distributed interval times for traffic bursts and Poisson distributed traffic inside the bursts. A Pareto distribution has the following probability density function:

$$P_{pareto}(x) = \frac{\alpha\beta^{\alpha}}{x^{\alpha+1}}, x \geq b \tag{3}$$

where α is a shape parameter and *b* is the minimum value of *x*. The mean value of a Pareto distribution is

$$E(x) = \frac{\alpha\beta}{\alpha - 1}, \alpha > 1 \tag{4}$$

For self-similar traffic α should be between one and two. The formula to generate a Pareto distribution is

$$X_{pareto} = \frac{b}{U^{\frac{1}{\alpha}}} \tag{5}$$

where *U* is uniformly distributed value in the range (0,1]. For the packet trains we used values 1 for *b* (minimum size of packet train) and 1.5 for the shape parameter α and for the silence periods between packet trains 8/1518 (size of the preamble compared to the maximum size of an Ethernet packet) for *b* and 1.2 for α.

A Poisson distribution, on the other hand, has the following probability density function:

$$P_{poisson}(x, \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}, (x = 0,1,...) \tag{6}$$

where λ is the shape parameter which indicates the mean value of the events in the given time interval. The formula for the Poisson cumulative probability function is:

$$F(x, \lambda) = \sum_{i=0}^{x} \frac{e^{-\lambda} \lambda^i}{i!} \tag{7}$$

The Poisson percent point function does not exist in simple closed form like for the Pareto distribution above. Hence, it is computed numerically and is only defined for integer values of *x*.

6 Network models

Communication networks are usually divided into two main categories: *connection-oriented* or *circuit-switched networks* and *connectionless* or *packet-switched networks*. Circuit-switched networks operate by forming a dedicated connection or circuit between the end points, *i.e.*, users have *reserved channel and resources* during the connection. Users can transmit datagrams without address information. Hence, events and duration times of sequential events as well as failure rates, delay profiles and jitters are very well known beforehand within a certain limit. The advantage of the circuit switched networking lies in its guaranteed capacity. The other advantage is the lower overhead in transmitted data. Addresses and other control information are not required to enclose into all the data packets, which might be very significant advantage in fraudless wireless environment. The disadvantage is reserved channel and bandwidth even if it not fully used, like in the normal phone conversations. However, especially for the wireless medium, discontinuous transmission and voice activity detection techniques for phone conversation has been developed in order to decrease the disadvantage.

In packet-switched networks, data is divided into small packets or datagrams which are multiplexed and transferred across the network in a high capacity connections, *i.e.*, several users *share the same channel and resources*. Intermediate nodes like routers in the Internet recognize received message types and after reading included address fields forward these to the next intermediate node or to the final destination. The final destination accepts the datagram targeted to it by interpreting included physical address field in the datagram header. The quality of service and reliability can not be guaranteed in the similar way than in circuit

switched solutions and probabilities as well as duration times of various events vary.

Parallel events on the different branches also increase uncertainty. Branches may cross each other, *e.g.*, on the routers or they can share partly the same communication paths like sharing the same link between the sequential routers. If they run, *e.g.*, into the same router then depending on the architecture of router the effect is either decreased processing time for each of the traffic flow or there is no effect at all due to separate processors and input/output gates for separate communication paths on the router. The situation is more complicated if traffic flows are combined to the same output branch producing higher traffic density on the branch and on the input of the next router. This may cause congestion and increased packet loss rate, too. Therefore, moderate traffic load on separate network branches may cause congestion and increased packet loss rate on the shared branch. However, computational consequence for the convergence time definition is only increased average delay and variance. Hence, methodologically the effects of parallel communication paths of the network can be modeled as Markov chain likewise in the case of sequential path.

In wireless networks parallel events around the communication path have always additional interference effects. Increased interference level increases packet loss rate which jointly increases average delay and variance. Also in this case the computational consequence for the convergence time definition is increased average delay and variance.

7 Example Network Models

Example communication network models, which carry Pareto-Poisson distributed Internet traffic were developed for illustrations of numerical evaluation of the convergence time definition. The first example considers fixed wireline network model with widely used ANSI C *rand()* random number generator. The second example presents the same network model with more sophisticated random number generator *ranC()* described above in section *Random numbers*. The third and fourth examples consider more complicated fixed wireline network models. A reference output variable on the examples was an *overall delay*. The overall delay consist of propagation and processing delays along a communication path. The average data rate for the models was either 12 Mbps, 60 Mbps, 120 Mbps or 600 Mbps. In this paper, a simulation was considered to be convergenced if the difference between the minimum and maximum total delays in a series of simulations was less than 0.25 % from the overall delay and the maximum value of the total delay do not differ more than 0.15 % from the overall delay. The overall delay was defined by simulating so long time than the output value did not change practically at all. The size of series of simulations was 10 runs per data rate.

The fifth example considers an ad hoc network model, in which the development of throughput as a function of simulation time is illustrated. In this case a simulation was considered to be converged when capacity fluctuated within 1% limits.

7.1 Wireline network models

In the first example the communication path from a source to a destination included a transmitter, 8 intermediate units, like routers, repeaters and multiplexers, and a receiver (Figure 2). The delay times were normally distributed in six (6) units and uniformly distributed in (2) unists. In Figure 2, for instance, the notation *normal 15:10* refers normal distribution with an expected time 15 ms and a variance 10 ms. The notation *uniform 1:10* refers uniform distribution with a minimum value 1 ms and a maximum value 10 ms. The probagation delays between the intermediate nodes were constants and not considered separately.

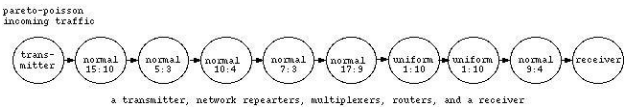

Fig. 2. The series of nodes from a transmitter to a receiver.

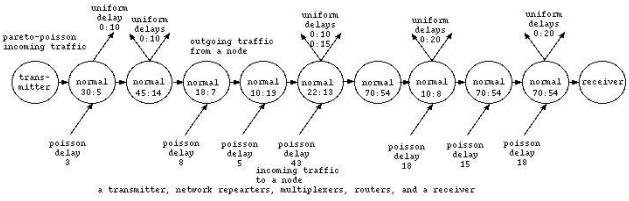

Fig. 3. The series of nodes from a transmitter to a receiver.

Table 1. Data rates (Mbps) and convergence times.

Data rate	12	60	120	600
Example 1	20 000	15 000	15 000	15 000
Example 2	15 000	10 000	5 000	500
Example 3	450 000	45 000	40 000	20 000
Example 4	50 000	15 000	10 000	2 000

Table 1 presents convergence times for different data rates on the different examples. In the first example convergence time sets after 15 000 ms for all the data rates except 12 Mbps. This seems quite a suprising because with the higher data rate there was more random sampling events in the model and the convergence time should be shorter, too. The same convergence time for the most of data rates may due to Pareto-Poisson traffic model, which may alone require 15 000 ms time to converge. This may also indicate that the used random number generator *rand()* produces biased/correlated random samples because the convergence times dramatically get shorter with *ranC()* random number generator, see Table 1, Example 2. In the example 2 the convergence time decreased as a function of data rate. This is due to more frequent events in the network.

In the third example communication path from a source to a destination node consisted of 9 intermediate units (see Figure 3). The used random number generator was *rand()* (Example 3 in Table 1). Delay times of intermediate units were normally or uniformly distributed. Incoming traffic flows from other cross-connecting network branches to intermediate units on the communication path caused Poisson distributed delays to datagrams on the path. Outgoing flows from the intermediate units on the communication path, transmitted with uniform delays, caused uniform delays to datagrams. In the fourth example the difference for the Example 3 is a different random number generator, *ranC()*.

7.2 Ad hoc network

In an example AODV (Ad Hoc On Demand Distance Vector) ad hoc network simulations, the reference variable for the convergence time definition was throughput (Mbps). Development of the throughput as a function of simulation time with 10 active nodes and pedestrian mobility is presented in Figure 4. Radio technology, physical and link layers of the example network were modelled according to the IEEE 802.11b WLAN standard with system capacity of 11 Mbps. Simulation time in Figure 4 spanned from 100 s to 5500 s. It can be noticed from the Figure 4 that the convergence time of the model is around 2000 s. For the simulation of, *e.g.*, extra transceivers effects, the convergence time should had been redefined due to the increased number of degrees of freedom. The convergence time should be redefined also if, *e.g.*, routing algorithm is changed or CCK (Complementary Code Keying) modulation/coding is changed to higher performance PBCC (Packet Binary Convolutional Coding) modulation/coding in the physical layer of ad hoc transceivers based on IEEE 802.11b WLAN (Wireless Local Area Network) standard or if connection oriented transport layer protocol is changed to connectionless protocol.

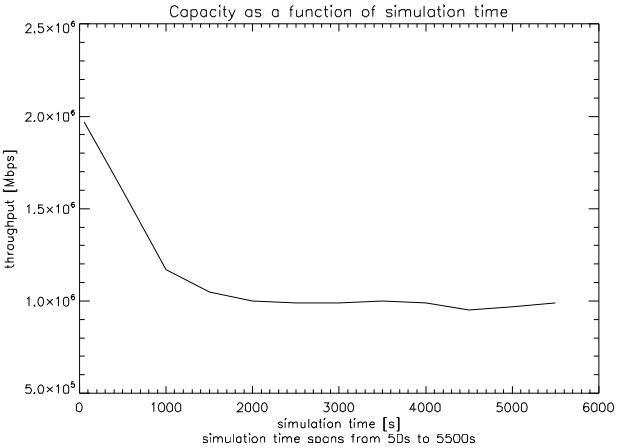

Fig. 4. Throuhgput of an ad hoc network.

8 Results

8.1 Results from wireline network models

The effect of an extra network element for the overall delay and variance in the Examples 1-4, when the simulation time ≥ convergence time is presented in Table 2. The extra element had normally distributed delay with an expected value of 22 ms and standard deviation of 11 ms.

We can notice from the Table 4 that the effect of the extra network element for the overall delay was indeed 22 ms in all the Example simulations, because oscillation limits for the overall delay are very narrow. With too short simulation times (100 ms) the overall delay oscillated much more around the average (Table 5). Variance level fluctuated also much more, when short simulation time was applied.

Table 2. Effect of a new element to the overall delay. Simulation time ≥ convergence time.

Example	st	delay	min del.	max del.
Example 1	20000	72.91	94.81	94.97
Example 2	10000	73.00	94.96	95.05
Example 3	40000	337.68	359.60	359.87
Example 4	10000	337.76	359.52	359.85

Table 3. Effect of a new element to the overall delay. Simulation time < convergence time.

Example	st	delay	min del.	max del.
Example 1	100	72.91	93.98	95.73
Example 2	100	73.00	94.48	95.35
Example 3	100	337.68	354.22	361.70
Example 4	100	337.76	359.06	362.12

Table 4. Effect of a new element to the overall delay and variance. Simulation time ≥ convergence time.

Examples	delay limits/ms	variance limits/ms
Example 1	-0.10 +0.06	-0.40 +1.295
Example 2	-0.06 +0.05	+2.16 +5.48
Example 3	-0.08 +0.19	-25.44 +134.62
Example 4	-0.24 +0.09	+14.97 +60.23

Table 5. Effect of a new element to the overall delay and variance. Simulation time < convergence time.

Examples	delay limits/ms	variance limits/ms
Example 1	-0.93 +0.82	-2.80 +3.56
Example 2	-0.52 +0.35	-10.03 +13.34
Example 3	-5.46 +2.02	-1233.54 +869.27
Example 4	-0.70 +2.36	-77.35 +167.78

8.2 Results from an ad hoc network model

Figure 5 presents throughput for data traffic of the ad hoc network as a function of simulation time when it spans from 10 seconds to 50 seconds [13]. It can be noticed that, *e.g.*, for capacity simulations if simulation time of 20 seconds is used, which is quite a typical value in numerous reported articles, absolutely too optimistic throughput value (more than 2.6 Mbps) is achieved. From Figure 4 it can be noticed than even if the simulation time is much longer than 20s but still too short, *e.g.*, 500 s, the average throughput of 1.6 Mbit/s is achieved. However, the real capacity value is around the 980 kbit/s, which is about 60% from the erroneous result achieved with too short simulation time.

Figure 6 presents average capacity of an ad hoc network as a function of number of nodes ranging from 5 to 50. For the 5 nodes average throughput was around 4.4 Mbps and for the 10 nodes it was only 980 kbps. This mainly dued to the increased number of packet collisions and transmission queueing time when the number of users grew. This is typical behaviour for a wireless systems with a random access based MAC (Media Access Control) channel division algorithm. In larger networks, for instance with 50 nodes, the link failures, packet collisions and transmission queueing become even more common and the network is rather congested. The average throughput for 50 nodes is only 92 kbps. This is very low for a local area ad hoc network with very limited coverage area and only pedestrian level mobility. It is very probable that with the higher level of mobility IEEE 802.11b WLAN standard based ad hoc networks will be fully congested due to increased packet error rate. Throughput value of 92 kbps may be enough for voice communication, poor level video transmission and other low data rate applications such as a short message delivery. It is definitely not enough for the fluent web browsing and email delivery except very temporarily, which are the most common applications of the WLAN front-end networks. Therefore, ad hoc networks with higher data rate applications should be based on, *e.g.*, 802.11a technology with admission control, which offers 54 Mbps system level capacity.

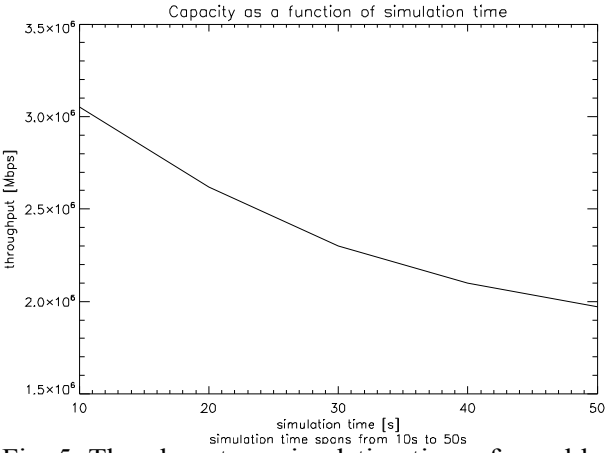

Fig. 5. Throughput vs. simulation time of an ad hoc network.

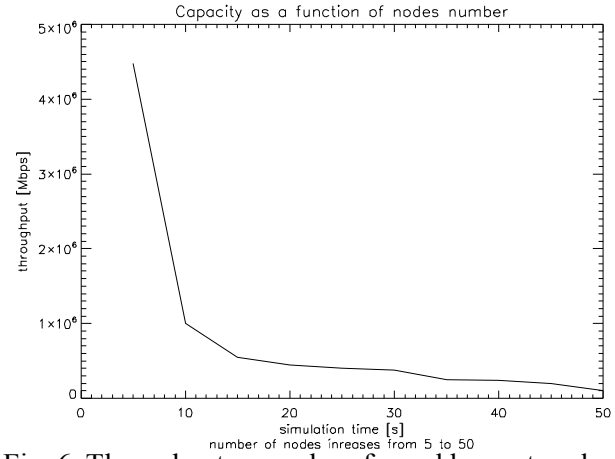

Fig. 6. Throughput vs. nodes of an ad hoc network.

9 Observations

Random sampling of stochastic distributions leads to an output value which depends on the number, types and parameters of distributions. For the modeling of distributed systems, like communication networks, one should also notice the effects of scalability to the convergence time because it is straightly dependent on the number of nodes in the model. The output value converges towards a constant value as a function of simulation time. Therefore, the influence of some parameter to the reference value can not be judged with simulation times shorter than the convergence time. In the case that the simulation time exceeds the convergence time, the effect of the

parameter under study can be seen on the output of the expected value and/or variance of the reference value. Deterministic phases increase the total expected value but do not change variance or convergence time. However, series of deterministic events may act as a stochastic distribution. From the results presented earlier we can instantly confirm that for the reliable calculation it is essential to define convergence time and let the simulation time exceeds it in order to research, *e.g.*, the effects of changed parameter.

It can also be deduced that random number generator design and implementation has an essential role to the convergence time. *RanC()* random number generator leads to much faster convergence times of the simulation model than *rand()* generator. Variance of the overall delay with *ranC()* generator is also higher than with rand generator. This may be indicate that *ranC()* utilizes single distributions more effectively with the full width of the distributions. In more complex systems simulations, the convergence time with the *ranC()* generator was about 4 times shorter than with the *rand()* generator. Therefore, the randomness of the overall delay with shorter simulation times than convergence times also caused higher errors with *rand()* generator than *ranC()* generator.

10 Conclusions

In this paper was described fundamentals of stochastic communication systems simulation, which were based on Markov chain theory and Monte Carlo method. A convergence time of the simulation model was defined for reliable simulations. Without the convergence time definition, it is possible that simulation results are randomly biased and erroneous.

The effects of scalability to the convergence time is straightly dependent on the degrees of freedom in the simulation model. Random number generator design is also in an important role because the faster the random number sampler setting time is the shorter the convergence time.

For communication networks like ad hoc networks with numerous nodes, moderate mobility of nodes and continuous transmission between the nodes, convergence time may be ramarkable. Hence, for example for reliable capacity estimates of ad hoc networks convergence time must be carefully defined in order to avoid erroneous and unrealistic values.

References:

[1] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, Applied Mathematics Series, Dover Publications, New York, 1964.

[2] K. Binder, Topics in Current Physics, Applications of the Monte Carlo Method in Statistical Physics, Springer-Verlag, Heidelberg, 1987.

[3] Bratley P., Fox B.L. and Schrage E.L., A Guide to Simulation, Springer-Verlag, 1983.

[4] B. Gaither, Empty Empiricism Empty Empiricism, Performance Evaluation Review, 18:2-3, 1990.

[5] F. Hartanao and W. Kreutzer and K. Pawlikowski and H.R. Siirisena, Quantitative Stochastic Simulation of Telecommunication Networks in DESC++, Computers Electronics and Engineering, 22:367-381, 1996.

[6] Knuth D.E., Seminumerical Algorithms, 2nd ed., vol. 2 of The Art of Computing Programming, Addison Wesley, 1981.

[7] W. E. Leland and M. S. Taqqu and W. Willinger and D. V. Wilson, On the Self-Similar Nature of Ethernet Traffic (Extended Version), IEEE/ACM Transactions on Networking, 2(1):1-15, 1994.

[8] A. M. Marsan and G. Balbo and G. Bruno and F. Neri, TOPNET: A toolfor the visual simulation of communication networks, IEEE Journal on Selected Areas in Communications, 8:1735-1746, 1990.

[9] Park S.K. and Miller K.W. Communication of the ACM, vol. 31, pp. 1192-1201.

[10] V. Paxson and S. Floyd, Wide area traffic: the failure of Poisson modeling, IEEE/ACM Transaction on Networking, 3(3):226-244, 1995.

[11] W. H. Press and S. A. Teukoisky and W. T. Vetterling and B. P. Flannery, Numerical Recipes in C, The Art of Scientific Computing, Cambridge Univesity Press, New York, 2nd edition, 1992.

[12] J. G. Proakis, Digital Communications, McGraw-Hill, New Jersey, 3rd edition, 1995.

[13] Taramaa M., “Capacity of Ad Hoc Networks”, Master’s Thesis, 2004, Oulu, Finland.

[14] W. Willinger and M. S. Taqqu and W. E. Leland and D. V. Wilson, Self-similarity in high-speed packet traffic: analysis and modeling of Ethernet traffic measurements, Statistical Science, 10:67-85, 1995.