Introduction to Network Simulator

Mouhamad IBRAHIM and Giovanni NEGLIA
mibrahim@sophia.inria.fr, gneglia@sophia.inria.fr

www-sop.inria.fr/maestro/personnel/Giovanni.Neglia/ns_course/ns_course.htm

Maestro team
INRIA Sophia-Antipolis - France
About Simulations

“The technique of imitating the behaviour of some situation or system (Economic, Mechanical etc.) by means of an analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel” (Oxford English Dictionary).

Approaches to simulations

- Discrete time systems models.
- Continuous systems models.
- Discrete events models.

NS is a discrete events simulator
- Calendar or Scheduler.
The simulator executes the event on top of the list. The execution of an event can generate other events that are placed in the list according to their time schedule. The simulation ends when the event \textit{End} is reached.
The simulator executes the event on top of the list.

The execution of an event can generate other events that are placed in the list according to their time schedule.

For example, the execution of event A has generated events D and E, event D is on top of the list (because \( t_4 < t_2 \)) and hence is the next event to be executed.

The simulation ends when the event \( End \) is reached.
nam trace

at the begin

    set nf [open out.nam w]
    $ns namtrace-all $nf

at the end

    $ns flush-trace
    close $nf

events trace

    set tracefile1 [open out.tr w]
    $ns trace-all $tracefile1

tracing a specific object

e.g. $ns trace-queue $n2 $n3 $file1
NS internal structure of a link:

Figure from *NS Simulator for beginners, E. Altman and T. Jiménez*
Trace (inside NS)

NS internal structure of a traced link:

Figure from NS Simulator for beginners, E. Altman and T. Jiménez
### Trace file format

Figure from *NS Simulator for beginners*, E. Altman and T. Jiménez

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
<th>From node</th>
<th>To node</th>
<th>Pkt type</th>
<th>Pkt size</th>
<th>Flags</th>
<th>Fid</th>
<th>Src addr</th>
<th>Dst addr</th>
<th>Seq num</th>
<th>Pkt id</th>
</tr>
</thead>
</table>
How can a machine generate a random number?

Random number generators in NS
- combined multiple recursive generator MRG32k3a
- set Rng1 [new RNG]
- Seeds $Rng1$ seed $a$
  - raw
  - pre-defined (independent streams divided in substreams),
  - heuristic seeds $Rng1$ seed 0
Streams and substreams

\[2^{191} = 3.1 \times 10^{57}\]

Stream \( i = 2^{191-127} = 1.8 \times 10^{19}\)

Substream \( j = 1\)

Stream Length = \(2^{127} = 1.7 \times 10^{38}\)

Substream Length = \(2^{76} = 7.6 \times 10^{22}\)

Substream \( j = 2\)

Stream \( i = 2\)

\[2^{31} = 2.3 \times 10^{15}\]

Substream \( j = 2^{51}\)
RVs from random generators

- Cumulative Distribution Function method
- Predefined random variables, e.g.
  ```
  set r1 [new RandomVariable/Uniform]
  $r1 use-rng $Rng1
  $r1 set min_ 0.0
  $r1 set max_ 10.0
  ```

- How to use
  - a predefined seed for each independent variable
  - a different substream for each independent run
  ```
  $r1 next-substream
  ```
When are RVs needed

- as a black-box model of aspects we prefer to ignore
- when the process to simulate is intrinsically stochastic
- when we want to average over different unknown initial conditions
  - but we could also look at steady state