4TC-PRF TP2	INSA Lyon - Departément Télécom
Simulation of a buffer	Marco Fiore

Summary: this TP is intended to let you familiarize with event-driven simulation. The goal of the TP is to understand:

- how to model a simple system into an event-driven simulator
- how to represent and postprocess the model outputs
- how to evaluate the impact of the inputs by running batches of simulations
- how to determine the correct settings of the system according to simulation results.

Additionally, you will further familiarize with *awk* and *gnuplot*, and learn about a few tricks useful to code an event-drivent simulator.

Keywords: *event-driven simulation, system modeling, performance evaluation, buffer dimensioning, simulator implementation.*

1 The system: a simple buffer

The system we will consider in this TP is the buffer represented in figure 1. There, packets arrive at the tail of a First In First Out (FIFO) queue representing the buffer, where they wait to be processed by a server (e.g., the CPU or the switching fabric). Since only one server is available, only one packet (that at the head of the queue) can be served at a time.

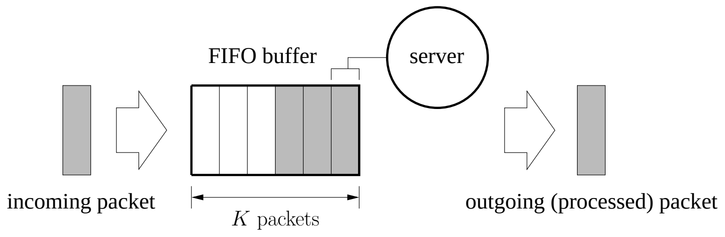

Figure 1: Graphical representation of a buffer of size K

We will assume that processing different packets does not take always the same amount of time: thus, we model the duration of packet services as a random variable S, exponentially distributed with parameter μ, i.e., $f_S(t) = \mu e^{-\mu t}$. The parameter μ is also referred to as the service rate, since it corresponds to the average number of packets served in the time unit (measured, e.g., in packets/s). Which is instead the average time to process one packet?

Q1.1

After the packet at the head of the queue has been processed, it leaves the buffer (e.g., it is passed to the lower/upper layer or it is transmitted over the channel) and all other packets shift towards the server. The new packet at the head of the buffer is then processed.

Note that the buffer has a size K, meaning that if a new packet arrives when K packets are already buffered, the new packet is discarded.

2 Planning the simulative performance evaluation

The layout of the performance evaluation process for this TP is presented in figure 2. You can recognize the structure we discussed in the course, and see the different choices in terms of modeling, input and output. However, the diagram in the figure is not complete, as we miss (i) the characterization of the inputs, (ii) the components of the event-driven simulator, and (iii) the decision on the outputs.

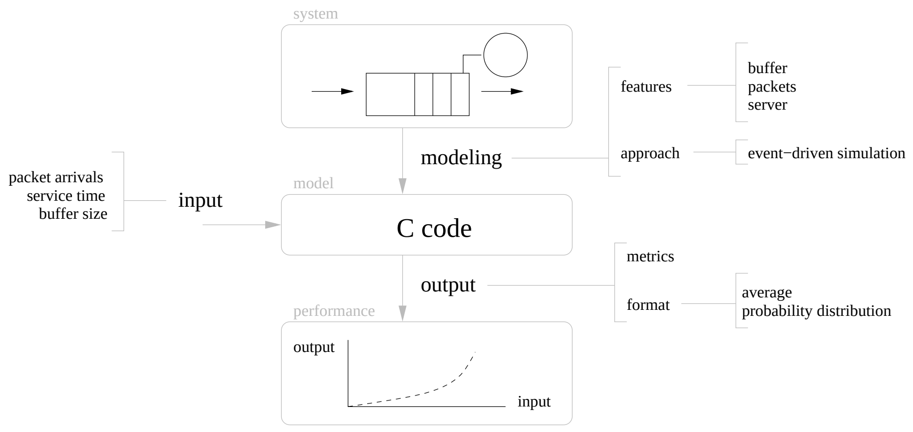

Figure 2: Performance evaluation process applied to our study

2.1 Input

The buffer size K, the nature of packet arrivals and that of service times are the three inputs that were chosen for the model. In simulation, this corresponds to the parameters to fed to the simulator.
We already said that service times are exponentially distributed, with parameter μ. For the moment, we will assume that the arrivals follow a Poisson distribution of parameter λt. Which is then the distribution of inter-arrival times? Which is the average time between the arrival of two subsequent packets?

Q2.1

Given the dynamic inputs identified above, the simulator is trace-based or stochastic? Why?

Q2.2

2.2 Simulator entities, state and events

The diagram in figure 2 does not include information on the fundamental components of the event-driven simulator, i.e., the entities, state and events.
The simulator entities basically correspond to the system features we decided to model. With reference to figure 2, our entities will be the buffer, the packets and the server (although, as we will see later on, the sever does not really need to be mapped to an simulation entity).
A meaningful system state consists of (a) number of packets buffered, and (b) the number of packets lost. As we will see, these two pieces of information are sufficient to characterize the simulation state at any given time instant.
Recall now that the events are the phenomena that change the system state. Which is your choice of events? Which change do they determine in the system state?

Q2.3

2.3 Output

The diagram in figure 2 does not specify which metrics should represent the output of the simulation. Which do you think are the important metrics to evaluate?

Q2.4

3 The event-driven simulator

An implementation of the buffer simulator is available for you to download on *moodle*, in the files `packet.[h,c]`, `fifo.[h,c]`, `fes.[h,c]`, and `buffer.c`. The first two couples of header/code files contain the implementation of the entities, the third couple implements the event structure and the Future Event Set (FES), and the last file contains the actual buffer simulator. Additionally, we will need a random number generator (RNG): we can use MINSTD, that we studied in TP1, and that is implemented in the file `rng.c`, along with functions to generate different random distributions.

You are not required to go through all these files. Just open the header files to see which are the commands to define and control the entities, events and the FES. Then look at the actual simulator implementation in `buffer.c`. This file contains the canvas of a discrete-event simulator of the buffer, that exploits the entities, events, FES and RNG implementations above. The code already has all the required includes and global variables. Also, it implements, in the `main` function, the command line processing routine, and the instructions for the FIFO buffer allocation and for the scheduling of the first event. Finally, you find the function `callback_departure()`, that processes the departure of a packet that has been served.
Write the pseudo-code of the event loop below, and to add it to the `main` function.

Q3.1

The departure is not the only event that can occur. Looking at the events you listed in your answer to **Q2.3**, write below the pseudo-code of the function(s) to process such event(s), and add the corresponding callback in `buffer.c`.

Q3.2

Once your functions are ready, debug the simulator (helpful debug functions are provided in `packet.h`, `fifo.h`, `fes.h`), then add tracing features: log all useful information in the trace file whose name is passed as the first parameter to the simulator. Which information are you tracing, and when?

Q3.3

In the following, we will consider interarrival and service times in the order of milliseconds (ms), thus pay attention to trace the system time with a microsecond (μs) precision.

Once the tracing features are active, perform some short tests to verify that the output trace file contains the data you expect. Also, have one of the teaching assistants validate your choice of events, implementation and tracing routines, so as to be sure that you have coded the right functions. We are now ready for some performance evaluation of the buffer.

4 Basic performance evaluation

Set the parameters so as to mimic the behavior of a server with exponentially-distributed service times and a mean of one packet served every millisecond. The buffer receives on average 900 packets per second with interarrival times exponentially distributed. The buffer can store up to 30 packets. Which are the values of λ, μ and K that you provide as input?

Q4.1

Let us run the simulation for a (simulated) time of one second, so that a few hundreds of packets arrive and are processed by the node. Study the output trace file, that you will name `tracefile.log`. Understanding the performance of the system from the raw data logged by the simulator is hard. Therefore, we postprocess the raw data so to obtain measures that "summarize" the log and provide a more intuitive representation of how the system performs.

The first metric we are interested in is the buffer occupancy, i.e., the number of packets in the system. As marked in figure 2, we want to express each metric in two formats: the average and the probability distribution. The average over time of the number of buffered packet is computed as

$$\bar{n} = \frac{1}{\sum_k \Delta t_k} \sum_k \Delta t_k n_k = \frac{1}{T} \sum_k \Delta t_k n_k$$

where $\bar{n}$ is the average number of buffered packets that we want to compute, and k is an event index: n_k is the number of packets in the buffer after event k, and Δt_k is the time between event k and the next event $k+1$ (that will change the number of queued packets to n_{k+1}). In other words, Δt_k is the time interval during which we have n_k packets in the buffer. Note that $\sum_k \Delta t_k$ is the time between the start of the simulation and the first event 1, plus the time between event 1 and event 2, plus the time between event 2 and event 3, etc., for all the events: it thus corresponds to the whole simulation duration T.

Download the awk script `postprocess.awk`, which performs the operation above. Try to understand how the script works. Then run it on `tracefile.log` to extract the average buffer occupancy. Now, try to run the simulations with shorter and longer durations, and compute the value of $\bar{n}$ for each run. Which is the effect of the simulation duration on the average number of buffered packets? Why?

Q4.2

In order to find a correct simulation duration, that provides stable results but that is not too expensive from a computational point of view, you can run simulations in batches. Download the shell script `buffered.sh`, and open it. Which operations does the script execute?

Q4.3

Run the script, and verify that the file it generates matches your intuition. The data can then be plotted using gnuplot, through the script `buffered.plt`. Download and run such a script. What do you observe? Which simulation duration you pick? Help yourself by changing the scale on the x-axis, if necessary. Since simulations are rather short, maintain a good margin.

Q4.4

Also, note that the image also portrays a straight line, marked as "Theory". Looking into the gnuplot script, you can see that this line represents the a constant value $\bar{n}_t$, obtained through the formula

$$\bar{n}_t = \begin{cases} \frac{\rho}{1-\rho} - \frac{K+1}{1-\rho^{K+1}}\rho^{K+1} & \text{if } \rho \neq 1, \\ \frac{K}{2} & \text{if } \rho = 1, \end{cases}$$

with $\rho = \lambda/\mu$. This is the average number of buffered packet predicted by the Continuous-Time Markov Chain that mathematically models the buffer. Note that this formula immediately gives the average buffer occupancy for any combination of λ, μ and K. You will learn to build and solve such a markovian model during the course: for the time being, the outcome of your simulation should match the formula.

5 Performance evaluation of the packet loss ratio

The number of queued packets gives us an interesting information on the utilization of the buffer, however a metric of more practical use is the loss ratio, i.e., the probability that a packet is dropped due to a full buffer. Add to `postprocess.awk` the lines to compute the loss ratio from the simulator raw output, and list them below.

Q5.1

When printing the packet loss ratio in the `postprocess.awk` script, pay attention to use a high precision (in the order of 10^{-9}), since packet losses may be rare events, resulting is extremely low (but non-zero) loss ratios. Also, use `loss` as the tag for the loss ratio statistic (in the same manner as, e.g., the tag `buffered` is used for the statistic on the average number of queued packets).

Write now a shell script named `loss.sh`, modeled after `buffered.sh`, that performs a batch of simulations for any combination of (λ,K), with:

- $\lambda \in \{500, 700, 900, 1000, 1100, 1300\}$ packets/s
- $K \in \{1, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100\}$ packets,

while the service rate μ stays the same as before. Configure the script so that it generates one log file `loss_`λ`.log` for each value of λ above. As an example, the file `loss_500.log` must contain on each line the system parameters followed by the loss ratio, for simulations with $\lambda = 500$ and any value of K.

Write then a gnuplot script named `loss.plt`, modeled after `buffered.plt`, that generates from all the logs `loss_`λ`.log` one figure named `loss.eps`. The figure must be contain multiple curves, and must be formatted as follows:

- on the x axis, the size of the buffer, K;
- on the y axis, the loss ratio. Since these values tend to span over different orders of magnitude, it is suggested to employ a logarithmic scale on the y axis. The formatting code is the following:
  ```
  set logscale y
  set yrange [1e-3:1]
  set ylabel "Packet loss ratio"
  set format y "1e{%L}"
  set mytics 10;
  ```
- one curve for each different value of λ.

Note that, to plot multiple curves in a single figure with gnuplot, it is sufficienct to provide to the `plot` command multiple data sets, separated by commas. You can use different point types (`pt` parameter of the plot command) to differentate the curves.

Once you have plotted the figure, draw below a *qualitative* representation of it.

Q5.2

Now look at the results: which is the general behavior of the loss ratio when increasing the buffer size? Do you notice any difference between the cases where $\lambda < \mu$ and those where $\lambda > \mu$?

Q5.3

How do you explain the behavior of the curves for $\lambda < \mu$?

Q5.4

How do you explain the behavior of the curves for $\lambda > \mu$?

Q5.5

Would you recommend the utilisation of a buffer when $\lambda > \mu$ in practice? Motivate your answer.

Q5.6

Through the analytical model of the buffer, we obtain that the theoretical probability that a packet is discarded is given by:

$$p_{loss} = \begin{cases} \frac{(1-\frac{\lambda}{\mu})\left(\frac{\lambda}{\mu}\right)^K}{1-\left(\frac{\lambda}{\mu}\right)^{K+1}} & \text{if } \lambda \neq \mu, \\ \frac{1}{K+1} & \text{if } \lambda = \mu. \end{cases}$$

Implement the formula above in the `loss.plt` script and plot the theoretical curves along the simulated ones. Do you have a match? How good it is?

Q5.7

Knowing this formula, can you explain the values of the packet loss ratio obtained in simulation for large values of the queue size K, when $\lambda > \mu$?

Q5.8

With reference to the case in which $\lambda < \mu$, we could be interested in understanding how large buffer must be, in order to guarantee a given packet loss ratio. Looking at `loss.eps`, which would be a sufficient buffer size, if the arrival rate is 900 packets/s, the service rate is 1000 packets/s, and the packet losses are required to be below 1%? And if the losses must be less than 0.1%?

Q5.9

Note that losses due to buffer overrun in the order of 0.1% are easily regarded as quite heavy in today's wired networks, and values one or two orders of magnitude lower Always considering the case of λ = 900 packets/s, Determine the buffer size needed to have a 0.001% packet loss ratio. What do you note?

Q5.10

6 Performance evaluation of the delay – BONUS

We just saw that increasing the buffer size can help to reduce the packet losses, which is obviously a desirable effect. However, increasing indiscriminately the buffer size risks to spoil another important metric, the delay, which corresponds to the time between the instant at which a packet arrives in the buffer and the instant at which it leaves it. Add to `postprocess.awk` the commands required to compute the average delay of packets that pass through the buffer, and output it under the tag `delay`. Pay attention to output the delay value with a sufficient precision, e.g., in the order of μs. List below the code you added in the body of the awk script.

Q6.1

Write now a shell script named `delay.sh`, modeled after `loss.sh`, that performs a batch of simulations for any combination of (λ,K) defined before, and the same service rate μ. Similarly to what done before, configure the script so that it generates one log file `delay_`λ`.log` for each value of λ above.

Write then a gnuplot script named `delay.plt`, modeled after `loss.plt`, that generates from all the logs `delay_`λ`.log` one figure named `delay.eps`. The figure must be similar to `loss.eps` above, but have the average delay on the y axis, instead of the loss ratio. The analytical model of the buffer provides the following formulation for the average delay experienced by packets:

$$\bar{d} = \begin{cases} \frac{1}{\mu(1-\rho^K)}\left(\frac{1-\rho^{K+1}}{1-\rho} - (K+1)\rho^K\right) & \text{if } \lambda \neq \mu, \\ \frac{K\lambda}{K+1} & \text{if } \lambda = \mu, \end{cases}$$

where $\rho = \lambda/\mu$. Add the analytical curves to your plot. Does they match your simulation results?

Q6.2

Once you have plotted the figure, draw below a *qualitative* representation of it.

Q6.3

Which is the effect of the buffer size on the delay? Which difference do you notice between the cases where $\lambda < \mu$ and those where $\lambda > \mu$?

Q6.4

How do you explain the behavior of the curves for $\lambda > \mu$?

Q6.5

How do you explain the behavior of the curves for $\lambda < \mu$?

Q6.7

Looking at the theoretical formula of the average delay, can you explain the values of the average delay obtained in simulation for large values of the queue size K, when $\lambda < \mu$?

Q6.8