UNIVERSITÀ DI PISA

Computer Engineering

Performance Evaluation of Computer Systems and Networks

Slotted random-access wireless network

Group Project Report

TEAM MEMBERS:
Tommaso Burlon
Francesco Iemma
Olgerti Xhanej

Academic Year: 2020/2021

Contents

1 Introduction

1.1 Problem Description

From the group project assignment:

*In a **slotted random-access network**, **N couples transmitter-receiver** share the same communication medium, which consists of **C separate channels**. Multiple attempts to use the same channel in the same slot by different transmissions will lead to collision, hence no receiver listening on that channel will be able to decode the message. Assume that each of the N transmitters generate packets according to an **exponential interarrival distribution**, and picks its channel at random on every new transmission. Before sending a packet, it keeps extracting a value from a **Bernoullian RV with success probability p** on every slot, until it achieves success. Then it transmits the packet and starts over. If a collision occurs, then the transmitter backs off for a random number of slots (see later), and then starts over the whole Bernoullian experiment. The number of backoff slots is extracted as $U(1, 2^{x+1})$, where x is the number of collisions experienced by the packet being transmitted.*

1.2 Objectives

The aim of the project report is the *Assessment of the Effectiveness of the Slotted Random-Access Network Protocol* described in the latter paragraph.

1.3 Performance Indexes

In order to define a metric of performance of the objective, the following Performance Indexes are defined:

- **Throughput**: let Tp be the Throughput to be measured, N_p the number of packets successfully sent to the corrisponding receiver, N_t the number of timeslot considered in the count of N_p, the Throughput can be measured as:

$$Tp = \frac{N_p}{N_t} \tag{1}$$

- **Response Time**: defined as the time that occurs from the first appearance of one packet at the Transmitter up to the reception of the packet at the Receiver.
- **Percentage of Deadlines not respected**: if we consider running this type of communication protocol of a real-time system with his own deadlines, there is a need to consider even this variable as a performance index.

2 Modeling

2.1 Introduction

We model the system as N couples of transmitter and receiver which communicate through C Channels. A collision can occur on a channel if more than one transmitters want to transmit a packet in that channel. The transmitter stores in a queue the packets that it wants to transmit and, then, it sends them; the channels "knows" if a collision occurs and handle it; the receiver only receive packets.

2.2 General Assumptions

The following general assumptions have been made:

- **Pure Slotted**: packets are attempted to be transmitter by the Transmitter only at the **beginning of the timeslot**
- **Constant Packet Size and Transmission Rate**: each packet has a constant packet size and each transmitter has a constant and equal transmission rate for which to transmit a packet (without collision) from the receiver to the transmitter will last **one timeslot**.
- **No Propagation Error in the channel**: the only cause of a failed transmission has to be considered as the packet collision. Other causes, i.e. path-loss, are neglected.
- **FIFO Queues of unlimited Capacity at the Transmitter**
- **Transmitters and Receivers always synchronized with the timeslot period**: in addition the receiver knows in which channel the transmitter will try to send his packet in each timeslot and the receiver will be ready to listen in the correct channel.
- **After an eventual collision the packet will change his channel choice**

2.3 Preliminar Validation

Before the implementation a preliminar validation phase is necessary to ensure that the model is correct. Let analyze if the assumptions made in the previous section are reasonable:

- The pure slotted assumption is reasonable due to the fact that exist some network protocols which work under this assumption, see for instance Slotted Aloha that is the most famous one.
- We consider the packet size constant because, in a network, there are small packet and huge packet, but if we want to consider the performance, then we have to take a mean length, otherwise it's possible that we consider too small or too large packets. Furthermore if the packet length is so large that more than one slots are needed, we can consider, from the viewpoint of the model, that this unique packet send in two different slots is like two packets of fixed-length send each one in a slot.
- The critical issue of every slotted network is the one related to the collisions: they have an huge impact on the general performance of the network, so it is reasonable to neglect the other propagation errors that are not network-specific or that depend from the environment (as path-loss).
- It's reasonable that the transmitter and the receiver are synchronized with the timeslot period because otherwise it would be very difficult every type of communications between the two entities. Moreover also slotted ALOHA requires a synchronization of this type.
- When a packet collide it's reasonable to think that the transmitter will change the transmission channel in order to avoid another collision. Indeed this along with the back-off time are the techniques that should avoid another collision.

2.4 Factors

The following factors have been defined which may affect the performance of the system:

- **N**: Transmitter-Receiver Couples.

- **C**: numbers of Channels.
- **p**: probability of success for sending a packet in the current timeslot for a Transmitter.
- *lambda*: exponential distribution rate of packets arrival at the Transmitter.
- $T_{deadline}$: deadline in case of communication with a real-time system.
- $T_{timeslot}$: timeslot duration.

3 Implementation

3.1 Modules

The following modules have been defined:

- **Transmitter**: duty of sending a packet to a specific channel. Parameters: *lambda*, p, packetsInQueue Signal (for the queue)
- **Channel**: duty of checking every channel in each timeslot for any collision. Parameters: $T_{timeslot}$, Throughput Signal
- **Receiver**: duty of receiving a packet from the channel. Parameters: $T_{threshold}$, Response Time Signal, Threshold Signal

3.2 Messages

A new format of message has been defined, in order to store all packet information,with the following fields:

- *simtime-t creationTime*: time in which the packet arrives for the first time at the Transmitter
- *int idChannel*: Channel choosed for the current transmission, may change in case of collision
- *int idTransmitter*: Index of the module Trasmitter related to the message
- *int indexTx*: Index of the gate in which the Trasmitter is linked to the Channel

3.3 Modules Behaviour

3.3.1 *Transmitter* Module Behaviour in the implementation

1. **Message Arrival** at the *Transmitter*:
    - **IF** an *ACK* has been received the packet at the top of the queue can be removed. **GOTO (2)**

- **ELSE IF** a *NACK* has been received, then the *Transmitter* starts his backoff time and **waits for another message**.
- **ELSE IF** an *Synchronization message* has been received **AND** the *Transmitter* is not in backoff-time **GOTO (2)**
- **ELSE IF** a packet arrives at the *Transmitter*, the Transmitter stores the packet in the queue, make a reschedule of the arrival of the next packet and **waits for another message**

2. The *Transmitter* tries to send the packet
   - **IF** success on the Bernoullian Experiment, then the packet will be forwarded to the *Channel*.
   - **ELSE waits for another message**

3.3.2 *Channel* Module Behaviour in the implementation

1. The *Channel* wakes up at the beginning of each timeslot and checks his channels status.
   - **IF** two or more packets have arrived in the same channel, the *Channel* will send a NACK to the *Transmitters* that have forwarded the packets in that specific channel.
   - **ELSE IF** one and only one packet has arrived in a channel, the *Channel* will send an ACK to the relative *Transmitter* and will forward the packet to the relative *Receiver*.
   - **ELSE** the *Transmitters* that did not send a packet will receive from the Channel a *Synchronization Message*

2. The *Channel* will gather of the packets for the current timeslot, to be processed in the next one.

   So this means that if the channel receives packets in timeslot j, then the informations about collisions are provided to transmitters in timeslot $j+1$. Hence, from the point of view of the transmitter, the information received at $j+1$ are referred to events took place in j

3.3.3 *Receiver* Module Behaviour in the implementation:

1. The *Receiver* wakes up when a packet arrives. Then are computed some statistics (concerning Response Time and Threshold).

4 Verification

In this section we present tests performed in order to verify our implementation.

4.1 Degeneracy Tests

In the degeneracy tests we verify the behaviour of our simulator with parameters set to 0 values. In all tests the simulator works properly. In particular the following observations can be inferred:

- If the number of channel is 0 then the simulation stops immediately because has no sense running a simulation with 0 channels.
- If the time slot size is 0 then the simulation doesn't stop and goes to infinite because on instant 0 time slots are continuously triggered.
- If the exponential mean is 0 then the simulation doesn't stop and continues to infinite because packets arrive at time 0 continuously.

4.2 Consistency Test

The consistency test verifies that the system react consistently with the output. In order to test this we perform two tests with the following parameters.

Test 1

- Number of transmitters: 1
- Number of channels: 500
- Send probability: 1
- Mean inter-arrival time: 10s (deterministic)
- Time slot size: 5s

Test 2

- Number of transmitters: 2
- Number of channels: 500
- Send probability: 1
- Mean inter-arrival time: 20s (deterministic)
- Time slot size: 5s

We expect that the result of the two tests are more or less equal because the behaviour of one source transmitting every 10 seconds must be similar to the behaviour of two sources transmitting every 20 seconds. We set the number of channels at 500 in order to neglect the effect of collisions.
The graph in figure 1 and in figure 2 show the results of the tests previously explained. We can see that the behaviour is very similar in both cases and so that the systems works as we expect.

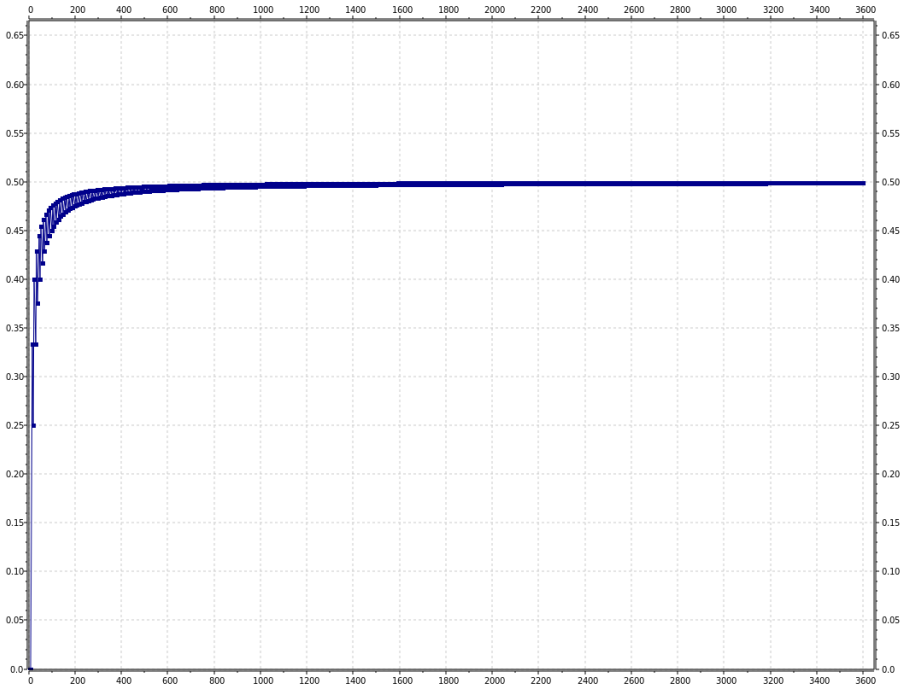

Figure 1: Consistency Test 1

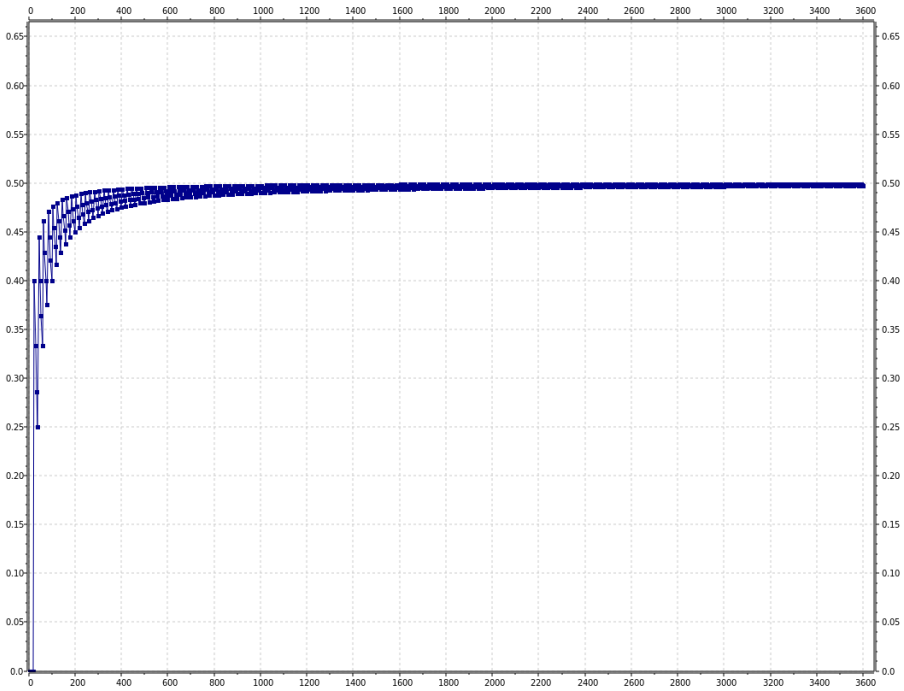

Figure 2: Consistency Test 2

4.3 Continuity Test

4.4 Test Simulations

5 Simulations Experiments

5.1 Calibration of Warm-Up Period and Simulation duration

5.2 Design of Experiments

5.3 Result Analysis

6 Conclusions