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 in the same timeslot, $T_{timeslot}$ the timeslot duration, the Throughput can be measured as:

$$Tp = \frac{N_{packetstimeslot}}{T_{timeslot}} = [s^{-1}] \tag{1}$$

- **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 Loss Packets**: due to the fact that the transmitter has a limited buffer capacity (see next chapter), there is a need to consider even this variable as a performance index.
- **Network Traffic**: (?)
- **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

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
- **FIFO Queues of limited Capacity at the Transmitter**
- **Transmitters and Receivers always synchronized with the timeslot period**
- **After an eventual collision the packet will change his channel choice**

2.3 Preliminar Validation

++++++++++++ **Aggiungere due righe** ++++++++++++++++

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.
- **K**: Transmitter Buffer Size.
- $T_{deadline}$: 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: K, $lambda$, p, overflow 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* ??
- *int indexTx* ??
- *int idReceiver* ??

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 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 tries to store the packet in the queue, make a reschedule 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 packet 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.

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

4.1 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