**~~SMART Objective~~**~~: Effectiveness of the Slotted Random-Access Network Protocol~~

**~~KPI~~** ~~(what defines the performances of the protocol):~~

* ~~throughput,~~
* ~~packet delay,~~
* ~~packets in buffer over time~~
* ~~network traffic~~
* ~~(compatibility with real-time Application = percentage of deadline not respected)~~

**~~Factors~~**~~:~~

* **~~N couples~~** ~~of Tx-Rx,~~
* **~~C number of channels~~**~~,~~
* ~~probability~~ **~~p~~** ~~(probability of sending a packet in the current timeslot),~~
* **~~lambda rate~~** ~~of the exponential distribution,~~
* ~~(~~**~~Deadline~~** ~~for real-time application)~~
* **~~time slot~~**

**~~Assumptions~~**~~:~~

* ~~Pure Slotted (packets can only be sent at the beginning of the time-slot),~~
* ~~One slot Packet Length,~~
* ~~No bit errors in the channel,~~
* ~~No trasmission delay in the channel (=> speed of an electromagnetic wave on air is very close to c. With a distance of 2km from the transmitter to the receiver, the transmission delay is in the order of some ns, so we choose to not considere trasmission delay)~~
* ~~FIFO queues with unlimited capacity (=> maybe M/M/1/C)~~
* ~~Tx and Rx always synchronized~~
* ~~After an eventual collision the packet will change its channel choice randomly~~

**~~Implementation in Omnet++~~**~~:~~

* **~~Transmitter~~**~~: campo nextArrivalTime per capire se arriverà pacchetto nel prossimo time slot. Campo verrà ricalcolato nel caso di più pacchetti nello stesso time slot~~
* ~~Channel: vector<PacketMsg\*> per capire se c’è stata collisione in un canale~~

**~~Implementazione protocollo~~**

**~~TRANSMITTER~~**

1. ~~Il~~ **~~Transmitter~~** ~~viene riattivato~~ **~~all’inizio di ogni timeslot~~** ~~da parte del~~ **~~Channel~~** ~~tramite dei messaggi di~~ **~~ACK~~**~~/~~**~~NACK~~** ~~(nel caso in cui il Transmitter abbia inviato un pacchetto nel timeslot precedente al Channel), oppure semplici~~ **~~messaggi di sincronizzazione~~**~~. Esso pertanto effettuerà le seguenti azioni:~~ 
   1. ~~Nel caso abbia ricevuto un~~ **~~ACK~~** ~~può cancellare dalla propria coda il messaggio in testa. Nel caso abbia ricevuto un~~ **~~NACK → backoff:~~** ~~il Transmitter non tenterà più di inviare pacchetti per un determinato tempo, ma continuerà a ricevere pacchetti da mandare al Channel.~~
   2. ~~Schedula gli istanti di arrivo dei nuovi pacchetti a~~ **~~partire dall’istante di ultima ricezione~~** ~~finchè l’istante di arrivo del nuovo pacchetto non supererà il tempo corrente (ovvero l’inizio del timeslot). Se gli istanti di arrivo sono precedenti al tempo corrente, I pacchetti verranno effettivamente inseriti (se lo spazio lo permette) nella coda.~~
   3. ~~Nel caso~~ **~~la coda sia non vuota e non sia in backoff~~**~~, il~~ **~~Transmitter~~** ~~proverà ad inviare il pacchetto in testa alla coda nel canale.~~
      1. ~~In caso di successo nel tentativo di trasmissione, il~~ **~~Transmitter~~** ~~invierà effettivamente il pacchetto al~~ **~~Channel~~** ~~ed attenderà una risposta da esso nel timeslot successivo.~~
      2. ~~Altrimenti ritenterà il rinvio al successivo time slot.~~
2. ~~Il~~ **~~Transmitter~~** ~~può essere anche risvegliato dagli~~ **~~arrivi dei pacchetti in qualsiasi istante~~**~~, il Transmitter effettuerà le seguenti azioni~~
   1. ~~Tenterà di memorizzare il pacchetto in caso ci sia spazio nel buffer~~
   2. ~~Schedulerà l’arrivo del prossimo pacchetto~~

**~~CHANNEL~~**

1. ~~Il~~ **~~Channel~~** ~~viene risvegliato all’inizio di ogni timeslot. Il~~ **~~Channel~~**~~, basandosi sulle sue strutture dati, invierà~~ **~~NACK/ACK,~~** ~~ai~~ **~~Transmitter~~** ~~che hanno inviato un pacchetto nello scorso timeslot, o un semplice~~ **~~messaggio di sincronizzazione~~** ~~nel caso opposto.~~
2. ~~ll~~ **~~Channel~~** ~~eliminerà I pacchetti ricevuti nei canali con collisioni e manderà I pacchetti “buoni” al~~ **~~Receiver,~~** ~~il tutto riferito al timeslot precedente.~~
3. ~~Il~~ **~~Channel~~**~~, riceverà tutti i pacchetti dai~~ **~~Transmitter~~**~~, per il timeslot corrente.~~

**~~Nota:~~** ~~Gran parte della comunicazione tra~~ **~~Transmitter e Channel~~** ~~avverrà sempre allo stesso istante (no propagation delay nei link), sfruttando messaggi con priorità.~~

**VERIFICATION EXPERIMENTS:**

**\*\*Modificare thoughput**

**1) Solo deterministici**

* **Un coppia tx-rx, un canale, Probabilità di trasmissione a 1, Interarrival time deterministico (> timeslot)**

**2) Scelta canale casuale**

* **Una coppia tx-rx, 5 canali, Probabilità di trasmissione a 1, interarrival time deterministico (con pacchetto sempre nel buffer : < timeslot)**

**3) P trasmissione casuale**

* **Un coppia, un canale, probabilità di trasmissione a 0.5, interarrival time deterministico (> timeslot) => modello con Binomial distribution per packet sent**

**3) Più coppie → collisioni possibili**

* **Due coppie, 1 canale, Probabilità di trasmissione a 1, interarrival time deterministico, buffer 5 slot => deve rispettare slotted aloha**
* **Tre coppie, 2 canali, Probabilità di trasmissione a 1, interarrival time deterministico, buffer 5 slot => deve rispettare slotted aloha**

**5) Test casi limite**

* **Degeneracy test (Probabilità trasmissione a 0, 0 coppie tx-rx**
* **Consistency test (1 coppia, 1 canale, interarrival deterministico a k vs 2 coppie 500 canali, interarrival deterministico a 2k)**
* **Continuity test**