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

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

* throughput,
* packet delay,
* percentage of loss packets (due to overflow in the queues)
* 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,
* Tx and Rx **buffer size**
* (**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 limited 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
* Eventi del Transmitter dovranno avere più priorità degli eventi del Channel, il canale entra in funzione quando tutti i transmitter hanno comunicato che vogliono eventualmente trasmettere un pacchetto
* Channel: vector<bool> per capire se c’è stata collisione in un canale

**Implementazione protocollo**

**TRANSMITTER**

1. Il **Transmitter,** nel caso in cui la sua **coda sia vuota** ed **arriva un nuovo pacchetto**, verrà riattivato all’inizio del prossimo time slot per provare ad inviare il pacchetto nel canale.
2. In caso il **Transmitter** venga riattivato (tramite un self message) all’inizio del timeslot successivo, esso:
   1. 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)
   2. Prova ad inviare il pacchetto in testa alla coda nel canale, se non ha successo, si rischedula un risveglio al prossimo time-slot, altrimenti vedi punto c).
   3. In caso di successo il Transmitter invia effettivamente il pacchetto al **Channel** ed attenderà una risposta da esso. I
      1. In caso di insuccesso → **backoff time**: ovvero il Transmitter continuerà a ricevere pacchetti ma non verrà più riattivato all’inizio del timeslot finchè il backoff time non sarà terminato.
      2. In caso di successo: se la coda non è vuota, si rischedula l’attivazione al prossimo time slot, altrimenti aspetterà un nuovo pacchetto e **tornerà al punto 1.**

**CHANNEL**

1. Il **Channel** viene “risvegliato” in caso di ricezione di un pacchetto da un qualsiasi **Transmitter** (nota, per come è pensato il transmitter, riceverà un pacchetto all’inizio di un timeslot). Il **Channel** si rischedulerà un risveglio allo stesso instante con priorità più bassa per ottenere tutti I pacchetti.
2. Una volta ottenuti tutti I pacchetti, dalle sue strutture dati sarà in grado di determinare le eventuali collisioni, mandando gli **ack** ed in **nack** ai **Transmitter** che hanno effettivamente inviato un messaggio.
3. Dopodichè il **Channel** eliminerà I pacchetti ricevuti nei canali con collisioni e manderà I pacchetti “buoni” al **Receiver**

**Nota:** Gran parte della comunicazione tra **Transmitter e Receiver** avverrà sempre allo stesso istante (no propagation delay nei link).