Report Computer Software - Challenge 3 by Eline Brader (s2674483), Lars Ran (s1403192)

Goal of this challenge was to obtain the highest possible efficiency (e) and fairness (f). Below the considered strategies will be discussed in their order of implementation.

Slotted ALOHA
The program was first run using just a simple slotted ALOHA strategy. This showed low e and an average f and was readily available from data provided.

Channel Turns
Slotted ALOHA strategy could be improved for fairness by creating a channel with a strategy somewhat equal to "*Time Division Multiplexing (TDM)*". Each client would have its own assigned slot in which it would be allowed to send data. This would however lead to many idle slots whenever a client would have nothing to send. With other clients possibly waiting to send at an idle slot, this would cause unnecessary losses in e. This could achieve f of 100 as each client would get a turn within an entire round in the order of their initial assigned client number.

Reset-On-Collision
The best strategy we were able to implement. Clients would have an initial assigned "clientNumber", but also a "turnOrder". The "turnOrder" would depend on several occurrences. N clients would be assigned an unique number from 1...N, which would always be the "clientNumber". Only at the beginning of the runs and whenever a collision would occur, "clientNumber" == "turnOrder". Clients would only have a "turnOver" number whenever these would have any data to send. If so, their pattern of sending would be similar to the one discussed in "*Channel Turns*". Example:

Let's start with N = 4 and for each client "clientNumber" == "turnOrder". If all clients would have something to send except for x (with x >=0 && x < 4), in all slots data will be sent except for that of clientNumber = x, which will return idle. The client causing the idle slot will be known by all other clients because of their allowed order to send. An idle slot will cause a reassignation of the "turnOrder", where all clients with clientNumber > x would get turnOrder--. If client x would have something to send again, this would always cause collision and all clients would be reset with "clientNumber" == "turnOrder". This pattern is repetitive until all data is sent.