**/\* Global Variables \*/**

**/\* total number of calls that were attempted \*/**

total\_calls = 0;

**/\* number of calls that were blocked at initialization \*/**

blocked\_calls = 0;

**/\* number of calls that got dropped during handover \*/**

dropped\_calls = 0;

**/\* initialize an array of 20 stations \*/**

stations = array(Stations, 20);

**/\* for each of the stations, assign an id and create 10 free channels \*/**

for i in 1 to 20 {

stations[i].available\_channels = 10;

}

**/\* Call Process \*/**

CallProcess() {

**/\* increase total number of calls attempted \*/**

total\_calls++;

**/\* generate base station idx using distribution X \*/**

curr\_station\_idx = random\_X();

curr\_station = stations[curr\_station\_idx];

**/\* generate car position (m) randomly in 0 to 2,000 (with equal probability) \*/**

car\_position = random\_unif\_continuous(0, 2000);

**/\* generate car direction randomly (with equal probability) \*/**

**/\* car\_direction = -1 if car moving left, and +1 if moving right \*/**

car\_direction = random\_unif\_discrete((-1, +1));

**/\* generate car speed (m/s) using distribution Y \*/**

car\_speed = random\_Y();

**/\* generate call duration (s) using distribution Z \*/**

call\_duration = random\_Z();

**/\* check if there is a channel available to make the call \*/**

if (curr\_station.available\_channels == 0) {

**/\* block the call at initialization \*/**

blocked\_calls++;

return;

} else {

**/\* acquire one of the available channels \*/**

curr\_station.available\_channels --;

}

**/\* calculate the distance (m) to the next station \*/**

if (car\_direction == -1) {

distance\_till\_station\_exit = car\_position;

} else {

distance\_till\_station\_exit = 2000-car\_position;

}

**/\* calculate the time (s) to the next station \*/**

time\_till\_station\_exit = distance\_till\_station\_exit / car\_speed;

**/\* while the call will continue as the station is exited \*/**

while (call\_duration > time\_till\_station\_exit) {

**/\* finish the journey in the current station \*/**

**Hold**(time\_till\_station\_exit);

**/\* update the call duration left \*/**

call\_duration -= time\_till\_station\_exit;

**/\* exit station and free up the channel \*/**

curr\_station.available\_channels++;

**/\* if the car is exiting the highway \*/**

**/\* at station 1 and exiting left, or at station 20 and exiting right \*/**

if ((curr\_station\_idx == 1 && car\_direction == -1) || (curr\_station\_idx == 20 && car\_direction == 1)) {

return;

}

**/\* handover the call to the next station \*/**

curr\_station\_idx += car\_direction;

curr\_station = stations[curr\_station\_idx];

**/\* check if there is a channel available to make the call \*/**

if (curr\_station.available\_channels == 0) {

**/\* drop the call during handover \*/**

dropped\_calls++;

return;

}

else {

**/\* acquire one of the available channels \*/**

curr\_station.available\_channels --;

}

**/\* the distance (m) to exit station will be the length of the station = 2,000 \*/**

distance\_till\_station\_exit = 2000;

**/\* calculate the time (s) to the next station \*/**

time\_till\_station\_exit = distance\_till\_station\_exit / car\_speed;

}

**/\* finish the journey in the current station \*/**

**Hold**(call\_duration);

**/\* terminate call and free up the channel \*/**

curr\_station.available\_channels++;

**/\* terminate the call process \*/**

return;

}