```
1
2  /*
3   *
4   * Simulation_Run of A Single Server Queueing System
5   *
6   * Copyright (C) 2014 Terence D. Todd Hamilton, Ontario, CANADA,
7   * todd@mcmaster.ca
8   *
9   * This program is free software; you can redistribute it and/or modify it
10  * under the terms of the GNU General Public License as published by the Free
11  * Software Foundation; either version 3 of the License, or (at your option)
12  * any later version.
13  *
14  * This program is distributed in the hope that it will be useful, but WITHOUT
15  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
16  * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
17  * more details.
18  *
19  * You should have received a copy of the GNU General Public License along with
20  * this program.  If not, see <http://www.gnu.org/licenses/>.
21  *
22  */
23
24 /*****************************************************************************/
25
26 #include <stdio.h>
27 #include "trace.h"
28 #include "main.h"
29 #include "output.h"
30 #include "packet_transmission.h"
31
32 /*****************************************************************************/
33
34 /*
35  * This function will schedule the end of a packet transmission at a time given
36  * by event_time. At that time the function "end_packet_transmission" (defined
37  * in packet_transmissionl.c) is executed. A packet object is attached to the
38  * event and is recovered in end_packet_transmission.c.
39  */
40
41 long
42 schedule_end_packet_transmission_event(Simulation_Run_Ptr simulation_run,
43 ,                                      double event_time,
44 ,                                      Server_Ptr link,
45 ,               Packet_Ptr packet)
46 {
47   Event event;
48
49   event.description = "Packet Xmt End";
50   switch (packet->source_id)
51   {
52     case DATA_PACKET:
53     event.function = end_data_packet_transmission_event;
54     break;
```

```
    case VOICE_PACKET:
    event.function = end_voice_packet_transmission_event;
    break;
    default:
    printf("Packet source id invalid.");
    break;
  }
  event.attachment = (void *) link;

  return simulation_run_schedule_event(simulation_run, event, event_time);
}

/*****************************************************************************/

/*
 * This is the event function which is executed when the end of a packet
 * transmission event occurs. It updates its collected data then checks to see
 * if there are other packets waiting in the fifo queue. If that is the case it
 * starts the transmission of the next packet.
 */

void end_data_packet_transmission_event(Simulation_Run_Ptr simulation_run, void *
link)
{
  Simulation_Run_Data_Ptr data;
  Packet_Ptr this_packet, next_packet;

  TRACE(printf("End Of Packet.\n"););

  data = (Simulation_Run_Data_Ptr) simulation_run_data(simulation_run);

  /*
   * Packet transmission is finished. Take the packet off the data link.
   */

  this_packet = (Packet_Ptr) server_get(link);

  /* Collect stats */
  data->number_of_data_packets_processed++;
  data->accumulated_data_packet_delay += simulation_run_get_time(simulation_run) -
  this_packet->arrive_time;


  /* Output activity blip every so often. */
  output_progress_msg_to_screen(simulation_run);

  /* This packet is done ... give the memory back. */
  xfree((void *) this_packet);

  /*
   * See if there is are packets waiting in the buffer. If so, take the next one
   * out and transmit it immediately.
   */
  if ((fifoqueue_size(data->data_packet_buffer) > 0) &&
(fifoqueue_size(data->voice_packet_buffer) == 0)){
```

```
108     next_packet = (Packet_Ptr) fifoqueue_get(data->data_packet_buffer);
109     start_transmission_on_link(simulation_run, next_packet, link);
110   }
111 }
112
113 void end_voice_packet_transmission_event(Simulation_Run_Ptr simulation_run, void *
    link)
114 {
115   Simulation_Run_Data_Ptr data;
116   Packet_Ptr this_packet, next_packet;
117
118   TRACE(printf("End Of Packet.\n"););
119
120   data = (Simulation_Run_Data_Ptr) simulation_run_data(simulation_run);
121
122   /*
123    * Packet transmission is finished. Take the packet off the data link.
124    */
125
126   this_packet = (Packet_Ptr) server_get(link);
127
128   /* Collect Stats */
129   data->number_of_voice_packets_processed++;
130   data->accumulated_voice_packet_delay += simulation_run_get_time(simulation_run)
    -
131   this_packet->arrive_time;
132
133   /* Output activity blip every so often. */
134   output_progress_msg_to_screen(simulation_run);
135
136   /* This packet is done ... give the memory back. */
137   xfree((void *) this_packet);
138
139   /*
140    * See if there is are packets waiting in the buffer. If so, take the next one
141    * out and transmit it immediately.
142   */
143   if(fifoqueue_size(data->voice_packet_buffer) > 0) {
144     next_packet = (Packet_Ptr) fifoqueue_get(data->voice_packet_buffer);
145     start_transmission_on_link(simulation_run, next_packet, link);
146   }
147 }
148
149 /*
150  * This function initiates the transmission of the packet passed to the
151  * function. This is done by placing the packet in the server. The packet
152  * transmission end event for this packet is then scheduled.
153  */
154
155 void
156 start_transmission_on_link(Simulation_Run_Ptr simulation_run,
157 ,                          Packet_Ptr this_packet,
158 ,                          Server_Ptr link)
159 {
160   TRACE(printf("Start Of Packet.\n");)
```

```
161
162   server_put(link, (void*) this_packet);
163   this_packet->status = XMTTING;
164
165   /* Schedule the end of packet transmission event. */
166   schedule_end_packet_transmission_event(simulation_run,
167          simulation_run_get_time(simulation_run) + this_packet->service_time,
168          (void *) link,
169    this_packet);
170 }
171
172 /*
173  * Get a packet transmission time. For now it is a fixed value defined in
174  * simparameters.h
175  */
176
177 double get_packet_transmission_time(void)
178 {
179   return ((double) PACKET_XMT_TIME);
180 }
181
182 double get_voice_packet_transmission_time(void)
183 {
184   return ((double) VOICE_PACKET_XMT_TIME);
185 }
186
187
188
```