```c

/*
 *
 * Simulation_Run of A Single Server Queueing System
 *
 * Copyright (C) 2014 Terence D. Todd Hamilton, Ontario, CANADA,
 * todd@mcmaster.ca
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the Free
 * Software Foundation; either version 3 of the License, or (at your option)
 * any later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
 * more details.
 *
 * You should have received a copy of the GNU General Public License along with
 * this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

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

#include <stdio.h>
#include "trace.h"
#include "main.h"
#include "output.h"
#include "packet_transmission.h"

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

/*
 * This function will schedule the end of a packet transmission at a time given
 * by event_time. At that time the function "end_packet_transmission" (defined
 * in packet_transmissionl.c) is executed. A packet object is attached to the
 * event and is recovered in end_packet_transmission.c.
 */

long
schedule_end_packet_transmission_event(Simulation_Run_Ptr simulation_run,
,                                      double event_time,
,                                      Server_Ptr link)
{
  Event event;

  event.description = "Packet Xmt End";
  event.function = end_packet_transmission_event;
  event.attachment = (void *) link;

  return simulation_run_schedule_event(simulation_run, event, event_time);
}

```

```
55  /*****************************************************************************/
56
57  /*
58   * This is the event function which is executed when the end of a packet
59   * transmission event occurs. It updates its collected data then checks to see
60   * if there are other packets waiting in the fifo queue. If that is the case it
61   * starts the transmission of the next packet.
62   */
63
64  void
65  end_packet_transmission_event(Simulation_Run_Ptr simulation_run, void * link)
66  {
67    Simulation_Run_Data_Ptr data;
68    Packet_Ptr this_packet, next_packet;
69
70    TRACE(printf("End Of Packet.\n"););
71
72    data = (Simulation_Run_Data_Ptr) simulation_run_data(simulation_run);
73
74    /*
75     * Packet transmission is finished. Take the packet off the data link.
76     */
77
78    this_packet = (Packet_Ptr) server_get(link);
79
80    /* Collect statistics. */
81    switch (this_packet->source_id) {
82      case DATA_PACKET:
83      data->number_of_data_packets_processed++;
84      data->accumulated_data_packet_delay += simulation_run_get_time(simulation_run)
    -
85      this_packet->arrive_time;
86      case VOICE_PACKET:
87      data->number_of_voice_packets_processed++;
88      data->accumulated_voice_packet_delay +=
    simulation_run_get_time(simulation_run) -
89      this_packet->arrive_time;
90    }
91
92    /* Output activity blip every so often. */
93    output_progress_msg_to_screen(simulation_run);
94
95    /* This packet is done ... give the memory back. */
96    xfree((void *) this_packet);
97
98    /*
99     * See if there is are packets waiting in the buffer. If so, take the next one
100    * out and transmit it immediately.
101    */
102
103   if(fifoqueue_size(data->buffer) > 0) {
104     next_packet = (Packet_Ptr) fifoqueue_get(data->buffer);
105     start_transmission_on_link(simulation_run, next_packet, link);
106   }
107 }
```

```
108
109 /*
110  * This function initiates the transmission of the packet passed to the
111  * function. This is done by placing the packet in the server. The packet
112  * transmission end event for this packet is then scheduled.
113  */
114
115 void
116 start_transmission_on_link(Simulation_Run_Ptr simulation_run,
117 ,                          Packet_Ptr this_packet,
118 ,                          Server_Ptr link)
119 {
120   TRACE(printf("Start Of Packet.\n");)
121
122   server_put(link, (void*) this_packet);
123   this_packet->status = XMTTING;
124
125   /* Schedule the end of packet transmission event. */
126   schedule_end_packet_transmission_event(simulation_run,
127          simulation_run_get_time(simulation_run) + this_packet->service_time,
128          (void *) link);
129 }
130
131 /*
132  * Get a packet transmission time. For now it is a fixed value defined in
133  * simparameters.h
134  */
135
136 double get_packet_transmission_time(void)
137 {
138   return ((double) PACKET_XMT_TIME);
139 }
140
141 double get_voice_packet_transmission_time(void)
142 {
143   return ((double) VOICE_PACKET_XMT_TIME);
144 }
145
146
147
```