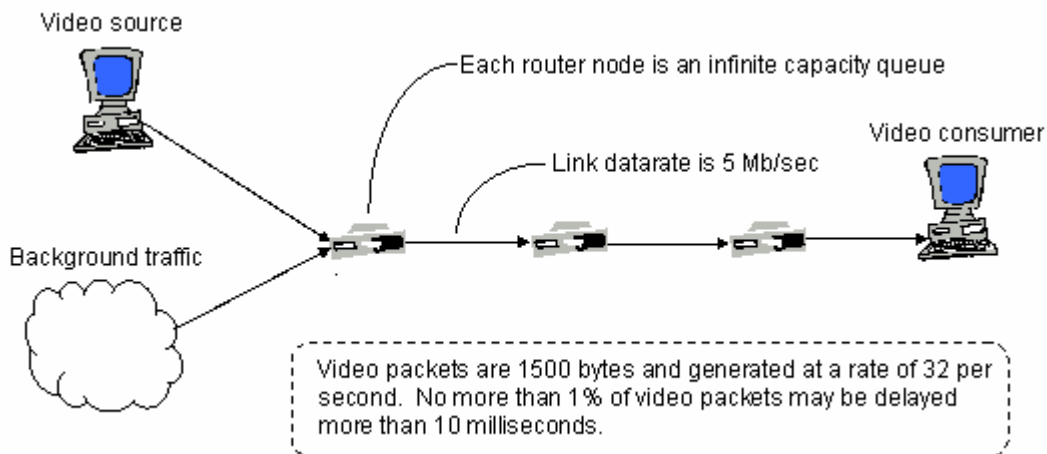


## Simulation (spring 2005) – Solutions to assignment #5

**Situation:** Video can be sent across a network in packets. Video traffic requires very low delay. We will assume that no more than 1% of packets delivered can have an end-to-end delay greater than 10 milliseconds. You have a three hop (three router) network with 5-Mbps links between the routers. You would like to know how much background traffic can be allowed on the network before video quality is compromised. Figure 1 shows your network, the video source, and video consumer.



**Figure 1** - Architecture of server showing job queue, CPU, and one disk

You have made measurements and determined that:

- Background traffic arrivals are Poisson.
- Background traffic packets are uniformly distributed between 100 and 1500 bytes in length.
- Video packets are generated deterministically at a rate of 32 per second.
- All video packets are 1500 bytes in length.

**Requirements:** You are to determine the maximum allowable rate of background traffic such that no more than 1% of video packets experience delay greater than 10 milliseconds. You are also to plot the mean video packet delay and percentage of video packets with delay more than 10 milliseconds for node utilization ranging from 15% to 95% in steps of 10%.

Submit a hardcopy and softcopy of your program and a short report (containing graphs!). In your report, discuss your results. That is, what do your results mean? The quality (correctness and neatness) of your source code will factor into your grade. Be proud of your high degree of craftsmanship!

**Extra Credit (10 points):** Determine a reasonable (i.e., implementable) way of minimizing the effect of reducing the effect of background traffic on video traffic and then evaluate your idea using your simulation model.

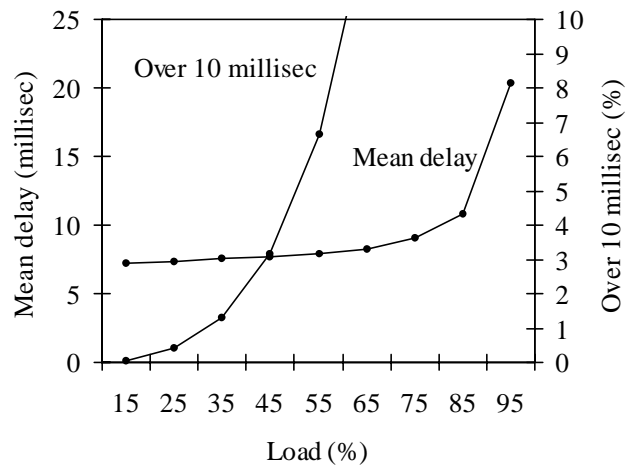
The CSIM simulation model for this problem is shown at the end of this solution. The video traffic offers 7.68% load to the nodes (i.e.,  $(32 \cdot 1500 \cdot 8) / 5000000 = 7.68\%$ ). The data traffic has a mean packet length of 800 bytes, so 100% offered load from data traffic only would be  $5000000 / (800 \cdot 8) = 781.25$  pkts/sec and so each 1% of offered load is 7.8125 pkts/sec. Figure 1 shows a graph of mean packet delay and percentage of packets delayed more than 10 milliseconds as a function of offered load (from 15% to 95% in increments of 10%). All points were run for 50,000 simulated seconds. It can be seen that 1% delay over 10 milliseconds occurs at about 30% offered load and then rapidly increases. Mean delay is fairly constant until about 85% offered load when it also increases rapidly.

The exact offered load at which 1% delay exceeds 10 milliseconds is very close to 32.3% (found by repeated executions of qos.c for different values of lambda).

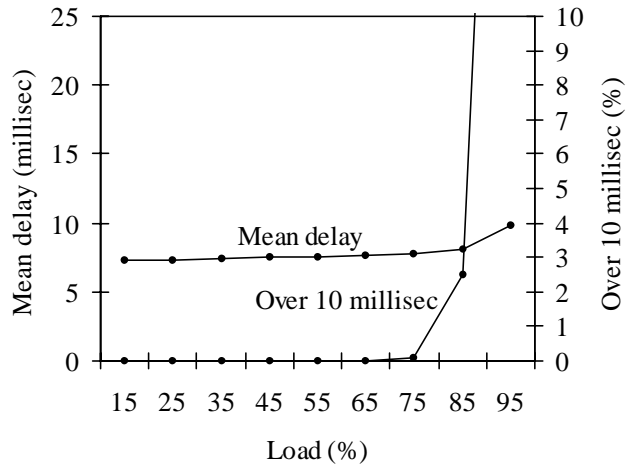
Extra Credit: A reasonable way to reduce the impact of the background traffic is to shape it. If the background traffic can be shaped to have deterministic, rather than exponential, interarrival time then the load at which 1% over 10 milliseconds delay can be achieved increases to about 81.8%. Also, the mean delay is less than 10 milliseconds for offered loads up to 95%. To model deterministic interarrival times we simply remove the “exponential” function call from the traffic generator function (easy!). An example of a company that makes traffic shaping products is Packeteer (see here: <http://www.packeteer.com>). This (traffic shaping) is currently a very “hot problem” (and a “hot money maker” too) as traffic load is increasing on campus and corporate networks. This extra credit portion of the assignment showed you the possible benefits from shaping. An open question (perhaps to be asked in a later assignment) is, what is the effect of shaping on the delay of the background traffic? Nothing comes for free!

---

.



**Figure 1.** Results for 3-hop network (no shaping)



**Figure 2.** Results for 3-hop network (WITH shaping)

```

//===== file = qos.c =====
//= A simulation of Video QoS for a three hop, 5-Mb/sec data rate network =
//=====
//= Notes: This program is the solution for assignment #5 for Simulation =
//= (Spring 2005). =
//-----
//= Build: Standard CSIM build =
//-----
//= History: KJC (03/11/05) - Genesis =
//=====
//----- Includes -----
#include <stdio.h> // Needed for printf()
#include "csim.h" // Needed for CSIM18 stuff

//----- Defines -----
#define BACKGROUND 0 // Background traffic stream id
#define VIDEO 1 // Video traffic stream id
#define DEADLINE 0.01 // Video packet deadline in seconds
#define DATA_RATE 5.0e6 // Link data rate in bits per sec
#define SIM_TIME 5.0e4 // Total simulation time in seconds

//----- Globals -----
FACILITY Node1_server; // CSIM facility for server in node #1
FACILITY Node2_server; // CSIM facility for server in node #2
FACILITY Node3_server; // CSIM facility for server in node #3
TABLE Delay_table; // CSIM table for job delay
int Depart; // Departure counter for video packets
int Over; // Over deadlien counter for video packets

//----- Prototypes -----
void gen_background(double lambda);
void gen_video(void);
void node1(double org_time, int pkt_len, int stream_id);
void node2(double org_time, int pkt_len, int stream_id);
void node3(double org_time, int pkt_len, int stream_id);
void consumer(double org_time);

//=====
//== Main program ==
//=====
void sim(void)
{
    double lambda; // Mean arrival rate (pkts/sec)

    // Create the simulation
    create("sim");

    // CSIM initializations
    Node1_server = facility("Node #1 server");
    Node2_server = facility("Node #2 server");
    Node3_server = facility("Node #3 server");
    Delay_table = table("Delay table");

    // Variable initializations
    Depart = Over = 0;

    // Parameter initializations
    lambda = 57.1875 + (8 * 78.125); // 57.1875 is 15% and 78.125 is 10%

    // Output begin-of-simulation banner
    printf(">>> Simulation is running (SIM_TIME = %f sec) \n", SIM_TIME);

    // Initiate generate functions and hold for SIM_TIME
    gen_background(lambda);
    gen_video();
    hold(SIM_TIME);
}

```

```

// Output results
printf("=====\n");
printf("==          *** CSIM video QoS simulation ***          == \n");
printf("=====\n");
printf("= SIM TIME              = %f sec          \n", SIM_TIME);
printf("= lambda                  = %f pkts/sec \n", lambda);
printf("=====\n");
printf("= >>> Simulation results          - \n");
printf("=----- \n");
printf("= Utilization node #1      = %f %%          \n",
    100.0 * util(Node1_server));
printf("= Utilization node #2      = %f %%          \n",
    100.0 * util(Node1_server));
printf("= Utilization node #3      = %f %%          \n",
    100.0 * util(Node1_server));
printf("= Mean video pkt delay     = %f millisec \n",
    1000.0 * table_mean(Delay_table));
printf("= Over 100 ms video pkts = %f %%          \n",
    100.0 * ((double) Over / Depart));
printf("=====\n");
}

//=====
//== Process to generate background traffic ==
//=====
void gen_background(double lambda)
{
    int      pkt_len;          // Packet length

    create("gen_background");

    // Loop forever to create background traffic packets
    while(1)
    {
        // Hold for a packet interarrival time (exponential)
        hold(exponential(1.0 / lambda));

        // Generate packet length
        pkt_len = random_int(100, 1500);

        // Send the packet to the first node
        node1(clock, pkt_len, BACKGROUND);
    }
}

//=====
//== Process to generate video traffic ==
//=====
void gen_video(void)
{
    int      pkt_len;          // Packet length

    create("gen_video");

    // Loop forever to create video packets
    while(1)
    {
        // Hold for a packet interarrival time (rate is 32 per second)
        hold(1.0 / 32.0);

        // Generate packet length
        pkt_len = 1500;

        // Send the packet to the first node
        node1(clock, pkt_len, VIDEO);
    }
}

```

```

}
}

//=====
//== Process for node #1 ==
//=====
void node1(double org_time, int pkt_len, int stream_id)
{
    create("node #1");

    // Reserve, hold, and release the node
    reserve(Node1_server);
    hold((8.0 * pkt_len) / DATA_RATE);
    release(Node1_server);

    // Send the packet to the next node
    node2(org_time, pkt_len, stream_id);
}

//=====
//== Process for node #2 ==
//=====
void node2(double org_time, int pkt_len, int stream_id)
{
    create("node #2");

    // Reserve, hold, and release the node
    reserve(Node2_server);
    hold((8.0 * pkt_len) / DATA_RATE);
    release(Node2_server);

    // Send the packet to the next node
    node3(org_time, pkt_len, stream_id);
}

//=====
//== Process for node #3 ==
//=====
void node3(double org_time, int pkt_len, int stream_id)
{
    create("node #3");

    // Reserve, hold, and release the node
    reserve(Node3_server);
    hold((8.0 * pkt_len) / DATA_RATE);
    release(Node3_server);

    // Send the video packets to the video consumer
    if (stream_id == VIDEO)
        consumer(org_time);
}

//=====
//== Video consumer ==
//=====
void consumer(double org_time)
{
    create("consumer");

    Depart++;
    record((clock - org_time), Delay_table);
    if ((clock - org_time) > DEADLINE)
        Over++;
}

```