1 Introduction

As described in the proposal for a project, the hybrid system we wish to simulate is the synchronized product of two hybrid automata: TCP congestion window automaton and a queue automaton. Both automata are shown in Figure 1, while their product is shown in Figure 2.

2 Topology and Assumptions

To determine the parameters in Figure 2, we rely on the following specific example, which is depicted in Figure 3. The link between the sender and the router has a bandwidth of 1Gbps with latency of 0.2ms. The link between the router and the receiver is a T3 line, i.e., its bandwidth is 45Mbps and its latency is 20 ms. The buffer at the router can accommodate up to 128 packets. All the packets are of equal size, which is 1500B. The following list defines the values for all of the parameters seen in Figure 2. Here, we assume that the round trip time (RTT) is constant over the entire transfer time.

- $B = 45\text{Mbps}$
- $L = 1500\text{B}$
- $q_{\text{max}} = 128$
3 Results and Conclusions

A Matlab\textsuperscript{®} script that uses an ODE solver with event catching is developed to plot the congestion window (cwnd) versus time in Figure 4 and the instantaneous queue size versus time in Figure 5. The source code is listed in the appendix of this document. From both these figures, it is apparent that once the queue size reaches its maximum capacity, packet drops occur and consequently the congestion window contracts to limit the sending rate.

- $ssh_0 = 256$
- $RTT = 0.04905\text{s}$
Figure 4: Matlab® output for cwnd.

Figure 5: Matlab® output for queue.
The plot of Figure 6 shows the simulation of the environment of Figure 3 using ns-2. Comparing Figures 4 and 6, one can see that both figures agree faithfully except the output from the Matlab® has slightly higher time-period. The reason for this is the assumption used in the Matlab® part that the RTT stays constant whereas ns-2 simulates the flow and the delay of packets realistically.

The output in Figure 4 is found to be very sensitive to the RTT value chosen in the Matlab® simulation. The value used previously was chosen to optimally arrive to same results of those of ns-2 and it was determined based on Figure 7 from the ns-2 output.
Figure 7: ns-2 output for RRT.
A Project.m

function varargout = Project(t,y,flag)

global ssh qmax B psize RTT

if nargin == 0
    flag = 'demo';
end

switch flag
    case ''
        varargout{1} = f(t,y);
    case 'events'
        [varargout{1:3}] = events(t,y);
    case 'demo'
        [varargout{1:5}] = demo;
    otherwise
        error(['Unknown flag'' ' flag '' .'']);
end

function dydt = f(t,y)

global ssh qmax B RTT psize

if y(4)==0,
    dydt = [log(2)*y(1)/RTT; 0; 0; 0];
elseif y(4)==1,
    dydt = [1/RTT; 0; 0; 0];
elseif y(4)==2,
    dydt = [log(2)*y(1)/RTT; (y(1)*psize/RTT-B)*1; 0; 0];
elseif y(4)==3,
    dydt = [1/RTT; (y(1)*psize/RTT-B)*1; 0; 0];
elseif y(4)==4,
    dydt = [1/RTT; 0; (y(1)*psize/RTT-B)*1; 0];
end;

function [value, isterminal, direction] = events(t,y)

global ssh qmax B RTT psize

if y(4)==0,
value = [y(1)-ssh; y(1)*psize/RTT-B];
isterminal = [1; 1];
direction = [1; 1];
elseif y(4)==1,
    value = [y(1)*psize/RTT-B];
    isterminal = [1];
    direction = [1];
elseif y(4)==2,
    value = [y(2); y(1)-ssh; y(2)-qmax*psize];
    isterminal = [1; 1; 1];
    direction = [-1; 1; 1];
elseif y(4)==3,
    value = [y(2); y(2)-qmax*psize];
    isterminal = [1; 1];
    direction = [-1; 1];
elseif y(4)==4,
    value = [y(3)-psize];
    isterminal = [1; 1];
    direction = [1; -1];
end
% ----------------------------------------------------------------------
function [tout,yout,teout,yeout,ieout] = demo

global ssh qmax B RTT psize

psize=1500*8;
ssh=256;
qmax=128;
B=45e6;
RTT=0.04905;
tstart = 0;
tfinal = 35;
y0 = [1; 0; 0; 0];
options = odeset('Events','on','OutputFcn','odeplot','OutputSel','[3]);

clf reset % deletes any stop button
set(gca,'xlim',[0 tfinal],'ylim',[0 1.5*psize]);
box on
hold on;
tout = tstart;
yout = y0.‘;
teout = [];
yeout = [];

end
ieout = [];  
for i = 1:30  
% Solve until the first terminal event.
[t,y,te,ye,ie] = ode23('Project',[tstart tfinal],y0,options);
if ~ishold
    hold on;
end  
% Accumulate output. This could be passed out as output arguments.
nt = length(t);
tout = [tout; t(2:nt)];
yout = [yout; y(2:nt,:)];
teout = [teout; te];           % Events at tstart are never reported.
yeout = [yeout; ye];
ieout = [ieout; ie];

% Set the new initial conditions
x = [y(nt,1); y(nt,2); y(nt,3); y(nt,4)];
if x(4)==0,
    if ie==1,
        y0(4)=1;  
        y0(1)=x(1);
    elseif ie==2,
        y0(4)=2;  
        y0(1)=x(1);
        y0(2)=0;
    elseif ie==3,
        y0(4)=4;  
        y0(1)=x(1);
        y0(2)=x(2);
        y0(3)=0;
    end
elseif x(4)==1,
    y0(4)=3;  
    y0(1)=x(1);
    y0(2)=0;
elseif x(4)==2,
    if ie==1,
        y0(4)=0;  
        y0(1)=x(1);
    elseif ie==2,
        y0(4)=3;  
        y0(1)=x(1);
        y0(2)=x(2);
    elseif ie==3,
        y0(4)=4;  
        y0(1)=x(1);
        y0(2)=x(2);
        y0(3)=0;
    end
elseif x(4)==3,
if ie == 1,
y0(4) = 1;
y0(1) = x(1);
elseif ie == 2,
y0(4) = 4;
y0(1) = x(1);
y0(2) = x(2);
y0(3) = 0;
end
elseif x(4) == 4,
if ie == 1,
ssh = x(1)/2;
y0(1) = 1;
y0(2) = 0;
y0(3) = 0;
y0(4) = 0;
end
end

% A good guess of a valid first timestep is the length of the last valid
timestep, so use it for faster computation.
options = odeset(options,'InitialStep',t(nt)-t(nt-1));

tstart = t(nt);