

TCP-FRIENDLY CONGESTION CONTROL MECHANISM FOR AN UDP-BASED HIGH SPEED RADAR APPLICATION AND CHARACTERIZATION OF FAIRNESS

Sangeetha L. Bangolae, Anura P. Jayasumana, V. Chandrasekar

Department of Electrical and Computer Engineering, Colorado State University, CO 80523, USA
E-mail: {sang, anura, chandra}@engr.colostate.edu

Abstract - Transfer of digitized radar data over the Next Generation Internet at data rates ranging from 64 Mbps to 384 Mbps, is an emerging real-time, high-bandwidth application using UDP transport protocol. We propose TCP-friendly Rate Adaptation Based on Loss (TRABOL) algorithm that relies on relevant QoS parameters feedback from the receiver to the source to control the transmission rate for congestion control. We also characterize the fairness of the system of UDP-based and TCP-based flows sharing a single bottleneck link bandwidth, using a throughput-based fairness index and show by experiments that the congestion-controlled radar application is TCP-friendly.

Keywords – high-bandwidth application, NGI, congestion control, TCP-friendliness.

I. INTRODUCTION

With the explosive growth of Internet, we witness the emergence of several high-bandwidth applications such as audio and video streaming, medical image transfer, and digitized radar data transfer. These network applications, to perform in real-time at a high data rate, rely on UDP, a non-congestion controlled protocol. TCP-based applications such as Telnet, FTP and web browsing constitute a majority of the Internet traffic and will continue to be so in the future. Hence, it is imperative for the emerging non-TCP based applications, especially those demanding high bandwidth, to be 'friendly' with these traditional applications. TCP-friendly behavior means that a real-time flow converges to fairness over the same timescale as that of a TCP session, under the same network conditions.

This paper addresses the congestion control and inter-protocol fairness issues with the help of an example *high-speed* application dealing with the transfer of digitized radar signals (DRS) over the Internet. VCHILL (Virtual CHILL) project [1] aims at a paradigm shift in radar and remote sensing research by exploiting the Internet technology to make the radar data remotely available to researchers and scientists in real-time and also facilitate virtual control of radar operation. UDP enables the real-time transfer of DRS over the Next Generation Internet/Internet2 [1]. However, to be TCP-friendly, it was vital to deploy an end-to-end congestion control for the application.

In this paper, we present the UDP-based radar data transfer protocol, discuss the implementation of the 'TCP-friendly Rate Adaptation Based On Loss' (TRABOL) algorithm, an end-to-end congestion control mechanism designed for the virtual radar application, the experiments

performed to demonstrate that the congestion-controlled radar application is TCP-friendly. We also define an index called throughput-based fairness Index for evaluating the TCP-friendliness of a system of TCP and non-TCP based flows.

II. VIRTUAL RADAR APPLICATION

The characteristics of importance of real-time radar data transfer application are summarized as follows:

- High-Bandwidth requirement for best operation; a high responsiveness to available bandwidth.
- Satisfactory operation with a minimum bandwidth threshold possible; yet increase in bandwidth provides a better display image.
- Tolerance to losses and end-to-end delay high, compared to audio and video streaming media.
- Smoothness not critical for proper functioning.

Figure 1 depicts the block diagram of the radar application showing transfer of DRS data over the NGI. The data generated by the radar is acquired by the data acquisition process of the DRS server and transferred over the NGI by the data transmission process using UDP protocol. At the remote end, the DRS client receives the data through the data receive process while the signal parameter estimation thread processes the data received and passes them to the graphical display unit for display to the end user.

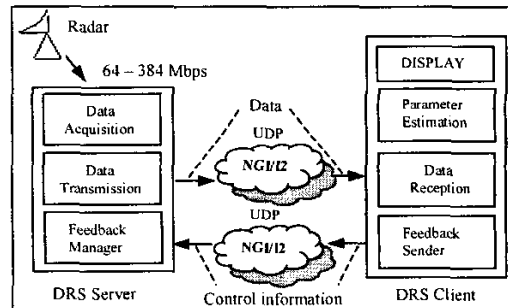


Fig. 1. Block Diagram of the process of Digitized Radar Data transfer over the Internet.

The control loop in the reverse direction aids in the implementation of the congestion control algorithm, which is based on the feedback sent by the receiver to the source about the quality of the data received. The radar image consists of rays, and the feedback is sent on a ray-by-ray basis. The data rate requirement is between 64 Mbps and 384 Mbps. For remote access, transfer of the data at this high data rate could only be supported by a backbone of

Next Generation Internet (NGI), which advocates gigabit speeds. NGI thus provides an opportunity to make the data remotely available and to reform the radar control operation to achieve a new geographically diverse interactivity [11].

III. CONGESTION CONTROL AND TCP-FRIENDLINESS

As the number of non-TCP based applications increases, the need for some form of end-to-end congestion control mechanism such as the one in TCP, has increased, in order to maintain the stability in the Internet. The idea of TCP-friendliness introduced congestion control of UDP-based transfers, especially audio and video streaming applications. The TCP-friendly equation which specifies the bandwidth of a TCP connection for a given packet size, MTU, round trip time (RTT), and loss rate (Loss) is given as,

$$\text{Bandwidth} = \frac{1.3 * MTU}{RTT * \text{Sqrt}(\text{Loss})} \quad (1)$$

The LDA (Loss-Delay Based Adjustment) algorithm [7] uses feedback from RTP/RTCP [10] for source data rate adjustment while RAP (Rate Adaptation Protocol) [9] uses AIMD (Additive Increase Multiplicative Decrease) policy for adjusting the transmission rate. [5] shows that a viable congestion control does not require TCP and for fairness of competing TCP traffic, it is feasible to use AIMD congestion control mechanisms that do not use a decrease-by-half reduction in response to congestion.

[2] proposes and evaluates a new class of non-linear congestion control algorithms called binomial algorithms. They also found that the TCP-friendliness equation [8] given by eq. (1) does not necessarily imply fairness relative to the performance of competing TCP traffic, especially for drop-tail bottleneck routers. A measure of fairness, Fairness Index, f is given by [6] is,

$$f = \frac{\left(\sum_{i=1}^n x_i \right)^2}{n \sum_{i=1}^n x_i^2} \quad (2)$$

This index gives a quantitative measure of the fairness of resource sharing and allocation problems in a distributed system. Here, x_i represents the resource allocation for user 'i' and the number of users sharing the resources is 'n'. However, this definition is inadequate, since it does not take into account the subscribed information rate or the user-level bandwidth requirement.

IV. THE TCP-FRIENDLY RATE ADAPTATION BASED ON LOSS (TRABOL) ALGORITHM

According to TRABOL algorithm, the server's transmission rate is continuously adjusted, in a TCP-friendly manner, based on losses in the network. When there is congestion, the server backs off aggressively and probes for any available bandwidth once congestion is not noticed. The sending rate increase and decrease policies are applied according to AIMD algorithm, which has been shown to efficiently converge to fairness [4].

TRABOL is a source-based rate control mechanism similar in its objective to the popular congestion control mechanism found in TCP, wherein the application-level feedback message received from the destination about loss rate decides the transmission rate of the server. The design criteria for severe-AIMD could be summarized as:

1. Decrease the sending rate aggressively upon congestion, by D-factor.
2. Increase the sending rate aggressively when the congestion indication returns a 0, by I-factor.

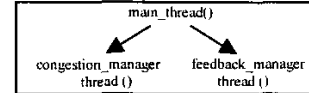


Fig. 2. Principal threads of the DRS Server

```

feedback_manager ()
{ /*Receive the feedback packet and calculate the loss rate*/
  receive_feedback ();
  Lr = (Pa - Pr) / t;
  /*Update the global buffer with the loss rate*/
  update_global_buffer (); }

```

Fig. 3. Feedback manager thread

```

congestion_manager () {
  if (Lr <= MIN_LOSS) {
    if (set_congestion == 0) {
      make_packet(); /*continue to send packets at the same rate*/
      send_packet();
      Rs = Pr / Ti; }
    else if (set_congestion == 1) {
      /*Apply Increase policy for a decrease policy before*/
      /*Increase the rate until TARGET_RATE or loss event occurs*/
      for (i=0; i < step_iteration; i++) {
        call_delay(Ga, k); /*Decrease the inter-group gap*/
        /*Increase the number of packets being sent by 1*/
        if (Pan != Pa)
          Pan = Pan + 1;
        for (i=0; i < iterations; i++) {
          make_packet(); /*Do nothing but maintain the rate*/
          send_packet();
          Rs = Pr / Ti; }
        }
      set_congestion = 0; }
  }
  else if (Lr > MAX_LOSS) {
    /*Reduce the rate by D-factor to MIN_RATE*/
    set_congestion = 1;
    Ti = Pr / Rs;
    Pan = Rs * Ri; /*Calculate the new number of packets to be sent*/
    Ta = Tr / Pan;
    Tr = Pm / Rm;
    Gi = Ti - Tr; /*Calculate the inter-group gap*/
    Ga = Gi / Pan;
    /*Send the new number of packets with inter-group gap*/
    make_packet ();
    send_packet ();
    call_delay(Ga, n); }
}

```

Fig. 4. Congestion manager thread

The TRABOL algorithm could be explained with the help of figures 2, 3 and 4. The main function as indicated in fig. 2 has two parallel threads namely, congestion_manager and feedback_manager.

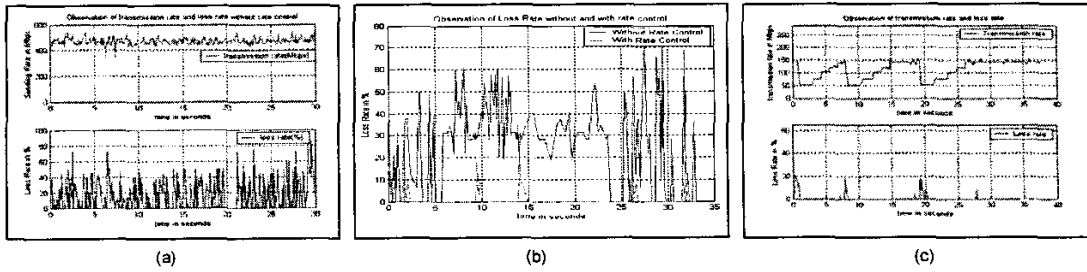


Fig. 5. (a) Sending Rate and Loss Rate at the destination with no rate control. (b) Loss rates at the destination with and without rate control. (c) Transmission rate (Mbps) and Loss rate (%) of the UDP-based radar application with rate control, D-factor = 0.45 and Initial Increase Rate = 20 Mbps.

Let P_n = number of packets sent in a ray of data, R_t = target rate, R_m = minimum rate, P_{nn} = number of packets sent during congestion, T_a = average time spent in sending a single packet at target rate, T_r = time spent in sending the reduced number of packets, G_t = time difference between T_a and T_r , G_a = average gap between two packets in a group, L_r = number of packets lost, P_r = number of packets received, n = number of packets in a group before a delay and R_s = sending rate.

Packets are formed and sent to the receiver over the network and the receiver upon receiving the packets in a ray sends a feedback message to the sender. The feedback manager collects and processes feedback information, while the congestion manager applies the specific TRABOL algorithm based on severe-AIMD. ' L_r ' represents the loss rate calculated at the sender based on the feedback received. If ' L_r ' is greater than the maximum loss rate, MAX_LOSS, then the sending rate is decreased aggressively by D-factor. This drop occurs during the beginning of the 'congestion cycle' and it is almost definitely followed by an increase policy determined by the I-factor. If ' L_r ' is less than or equal to the minimum loss rate, MIN_LOSS, it is checked if there was a congestion event (characterized by a decrease policy) earlier. A decrease policy is followed by an increase policy given by I-factor; otherwise the rate is maintained constant.

We used an Initial Increase Rate (IIR) of 20 Mbps. Then, I-factor is given by,

$$I = IIR * (1 - L_r) \quad (3)$$

If the current rate of transmission is R_{old} , then the new rate is given by,

$$R_{new} = R_{old} + I \quad (4)$$

The decrease and increase policies are applied by either increasing or decreasing, the gap between groups of packets in ray and the number of packets sent in a ray. The number of iterations to remain at a step (after an increase policy is applied), given by 'iteration', is determined by the loss rate while the number of steps, given by 'step_iteration', is determined by the target_rate or a loss event occurrence. Using the severe-AIMD approach, it can be shown that a neighboring TCP flow will get a fair share of bandwidth if it exists over a time period at least equal to one 'congestion epoch'. Thus we can argue that this end-to-end congestion

control algorithm is feasible, since it ensures convergence to fairness and efficiency at least for certain timescales.

V. PERFORMANCE EVALUATION AND RESULTS

In this section we discuss the experiments conducted to analyze the performance of the proposed congestion control scheme. Feedback packets are sent from the receiver to the source, after the receipt of every ray; approximately, about every 120 ms. The design of the test bed consists of a sender and a receiver connected through a single gigabit link on the LAN. The maximum transmission rate attained on this link is about 650 Mbps for UDP-based data transfer. Gigabit data rates provided by the NGI will thus make the Virtual CHILL project realizable in the future Internet. For experiments, the radar operation was emulated by reading the radar data from the shared memory with fixed time spacing.

Figure 5(a) shows the server rate in Mbps and the loss rate in % as a function of time for the case when there is no rate control at the server end. The server throttles the receiver and the network by sending data at the maximum rate of about 500 Mbps. At the destination, we perceive a fast moving poor quality radar image. Swamping of the Internet resources occurs since there is no congestion control. The vast difference in the loss rates between the cases when there is rate control and when there is no rate control is illustrated in fig. 5(b). With rate control, the loss rate consists of short living peaks not exceeding 25%, but for the initial transient.

In fig. 5(c) we show the transmission rate of the server in Mbps and the corresponding loss rates in % as a function of time in seconds. The target rate was set as 150 Mbps and the minimum rate as 45 Mbps. MIN_LOSS was 5% and MAX_LOSS was 10%. The increase in the rate during a congestion cycle is done either until the target rate is reached or until the receiver reports losses. Thus, the server reaches a steady-state rate and operates at that rate unless losses are reported through feedback. Assuming a loss rate of 25%, and IIR of 20 Mbps, I-factor, $I = (1 - L_r) * 20 = 15$ Mbps. Based on the set target rate, minimum rate, available link bandwidth, the receiver capacity, and the loss rate, the characteristics observed would be different. An example case has been presented here.

VI. TCP-FRIENDLINESS - MEASUREMENTS

In this section we describe experiments performed using the configuration in fig. 6 to demonstrate TCP-friendliness of the application. An internal network of two different networks connected through a bottleneck link of bandwidth 92 Mbps was created. Let us consider two streams of traffic from A and C, going to B and D respectively during the same time period. Under ideal conditions, fair allocation of bandwidth between the two streams would mean that the bottleneck link bandwidth is shared equally as about 46 Mbps each. *Netperf* was used to determine the bandwidth utilized by TCP, between hosts A and B.

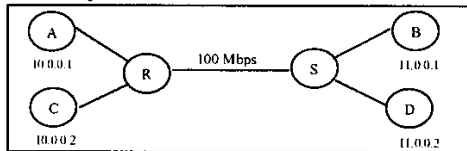


Fig. 6. Test-bed configuration

Throughout the experiments, a TCP stream was run between hosts A and B while a TCP or a UDP stream was run in parallel between hosts C and D so that the two streams shared the bottleneck link bandwidth and the bandwidth utilization by the TCP stream between hosts A and B is analyzed. *Tcptrace* was used to trace the response curves while *tcpdump* dumped the traffic on the network.

When both the streams are TCP-based, they share equal bandwidths of 46 Mbps each, because of the effective congestion control mechanism of TCP. Consider a continuous UDP stream using *netperf* running between hosts C and D. We observe in fig. 7(a), the throughput of the TCP traffic and infer that the average bandwidth share of the TCP stream does not exceed 8 Mbps. This was because the UDP application did not back off during congestion while TCP did and ended up with poor throughput. Thus non-congestion controlled UDP is highly unfair to competing TCP traffic. Fig. 7(b) shows the effect of introducing UDP as cross traffic. This plot shows that the throughput of TCP decreases when the UDP flow is started, and increases to a high value if the UDP traffic is removed.

Assume that the non-congestion controlled UDP-based radar application is run between hosts C and D. The UDP application sends huge bursts of data with gaps of few milliseconds. The throughput plot of the TCP stream

(between hosts A and B) in fig. 7(c) shows that TCP gets a low bandwidth share but during the gaps between the bursts of UDP data, TCP tries to send as much data as possible. The average throughput attained by the TCP stream was about 17 Mbps.

Consider the case when congestion controlled radar application was run between hosts C and D. Fig. 8 shows that the average throughput for the TCP stream obtained over a period of 18 seconds was about 43 Mbps. Thus we show that the UDP-based radar application has an efficient congestion control mechanism that makes it TCP-friendly.

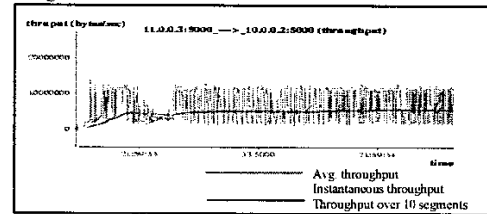


Fig. 8. Throughput of a TCP flow while sharing the bottleneck link with the rate-controlled radar application.

A. Throughput-based Fairness Index (TFI)

Equation (2) defines the index of fairness for resource allocation in shared computer systems [6] that could be applied in general to any system with resources to be shared among users. In Computer Networks, this fairness index is defined to quantify the fairness issues among flows that implement window based flow control. In this paper, we characterize a throughput-based fairness index for heterogeneous flows (TCP and non-TCP) sharing a single bottleneck link. Considering application-limited bandwidth conditions, it is neither necessary nor desirable for flows to have equal share of the available bandwidth.

We consider TCP-friendliness as a condition wherein a non-TCP flow does not hinder the performance of a neighboring TCP flow sharing the same bottleneck link bandwidth. During congestion, according to TFI, the bandwidths of the flows are reduced proportionately, while in an equal share allocation, all the bandwidths are reduced to give an equal share to all the flows.

For a two-flow case, let x_1 and x_2 represent the bandwidths of a TCP and a non-TCP flow sharing a single bottleneck link respectively, and let y_1 and y_2 be the

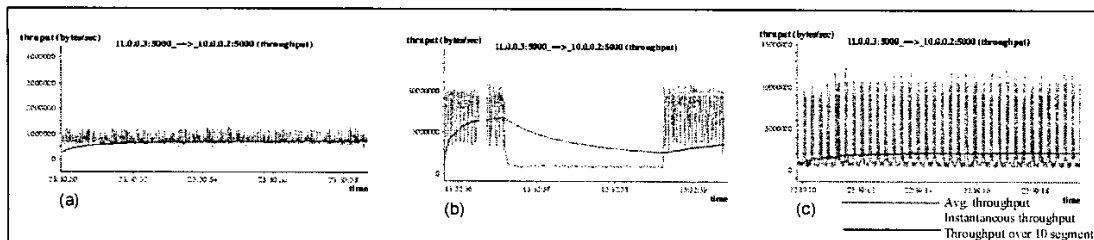


Fig. 7. (a) Throughput of a TCP flow while sharing the bottleneck link with an UDP flow. (b) Throughput of a TCP flow suffers when UDP is introduced as a cross traffic. (c) Throughput of a TCP flow while sharing the bottleneck link with the UDP-based radar application with no rate control.

corresponding bandwidths of these flows when it is the sole user of the link. If 'B' is the bottleneck link bandwidth, a proportionally fair allocation during congestion would be,

$$x_1 = B * \frac{y_1}{y_1 + y_2}; x_2 = B * \frac{y_2}{y_1 + y_2} \quad (5, 6)$$

This means that during congestion, all the flows sharing the same bottleneck link bandwidth get a proportional share of the bandwidth rather than an equitable share. From the above equations, it can be deduced that for fairness, x_1/y_1 and x_2/y_2 should be equal, which implies that the presence of the UDP flow affects the TCP traffic by the same factor as a TCP flow. For 'n' flow case, the proportional bandwidth share during congestion would be given as:

$$x_n = B * \frac{y_n}{\sum y_i} \quad (7)$$

The throughput-based fairness index of the system would be characterized as:

$$TFI = \frac{\left[\sum_{i=1}^n \frac{x_i}{y_i} \right]^2}{n \sum_{i=1}^n \left(\frac{x_i}{y_i} \right)^2} \quad (8)$$

We say that the system is perfectly fair, if TFI = 1. For flows that have equal bandwidth share requirement, the above equation for TFI reduces to equation (2).

Table 1
Average Throughput of a TCP flow (x_1) under various conditions

Flow x_2	Throughput of flow x_1	TFI
TCP	46 Mbps	1
UDP	8 Mbps	0.59
UDP _{unctrl}	17 Mbps	0.71
UDP _{ctrl}	43 Mbps	0.995

We summarize the results of the experiments in table 1, which shows the average throughput of the TCP flow and the TFI under four test conditions. Here, x_1 represents the TCP flow while x_2 represents the competing traffic introduced along the same bottleneck link with x_1 . UDP_{unctrl} stands for the non-congestion controlled radar application, while UDP_{ctrl} stands for congestion controlled, TCP-friendly radar application. We also evaluate the throughput-based fairness index (TFI) in each of the cases using equation (8). It can be seen that the radar application performs better with congestion control.

VII. CONCLUSIONS AND FUTURE WORK

This paper discussed the issues involved with implementation of non-TCP based high bandwidth data transfer applications and a means of deploying end-to-end congestion control. A TCP-friendly Rate Adaptation Based on Loss (TRABOL) mechanism was implemented for an emerging, high-bandwidth application such as digitized radar data transfer over the NGL. The implementation of this source based rate control mechanism using the AIMD approach was shown to make the application TCP-friendly. A key contribution of this paper is in characterizing the inter-protocol fairness based on a metric called throughput-

based fairness index that quantifies the fairness a non-TCP flow shows towards a competing TCP-based flow while sharing the bottleneck link bandwidth. We also argued that this bandwidth should be shared proportionately among all the flows rather than equally, during congestion. It was noticed that the congestion-controlled radar application had a fairness index higher than that without congestion control.

We are investigating on various QoS approaches to provide a better quality of service to the client. DiffServ Domain SLA (Service Level Agreement) used to keep updates of the network characteristics is an interesting area to explore. This will help us to negotiate and offer the rates according to the conditions of the network. The radar application represents a new class of emerging Internet applications that cannot function well over TCP. This class of applications differs from the streaming media in its requirements and we aim at generalizing the rate control algorithm and hence the TCP-friendliness constraint for all such applications.

ACKNOWLEDGEMENTS

This research was supported in part by the DARPA Next Generation Internet (NGI) program and the NSF Information Technology Research (ITR) Grant no. 0121546.

REFERENCES

- [1] S. Bangolae, "End-to-end Congestion Control Mechanism for Real-time High-bandwidth Applications: A Case Study with Digitized Radar Data Transfer", Master's Thesis, Colorado State University (In Progress).
- [2] D. Bansal, H. Balakrishnan, "TCP-friendly Congestion Control for Real-time Streaming Applications", MIT Technical Report, M.I.T. Laboratory for Computer Science, May 2000.
- [3] J. C. Bolot, T. Turlitti, "Experience with Control Mechanisms for Packet Video in the Internet" - INRIA, France: 1999.
- [4] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithm for congestion avoidance in computer networks. Journal of Computer Networks and ISDN, 17(1): 1-14, June 1989.
- [5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", SIGCOMM 2000, August 2000.
- [6] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems", DEC Research Report TR-301, September 1984.
- [7] D. Sisalem, H. Schulzrinne, "The Loss Delay Based Adjustment Algorithm: A TCP friendly adaptation scheme", Proceedings of NOSSDAV, Cambridge, UK: July 8-10 1998.
- [8] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control, Technical Note sent to end2end-interest mailing list, available from website, www.psc.edu/networking/papers/tcp_friendly.html: January 1997.
- [9] R. Rejaie, M. Handley, D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for real-time streams in Internet". In Proceedings of IEEE, INFOCOM 1999: March 1999.
- [10] H. Schulzrinne, S. Cassner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for real-time Applications; Internet Engineering Task Force, RFC 1889: January 1996
- [11] "Virtual CSU-CHILL National Radar Facility", Website: <http://www.engr.colostate.edu/ecs/Research/vchill/vchill.html>