

# Performance Evaluation of a Memory-Based TCP-friendly Rate Adaptation Algorithm for a Real-time Radar Application<sup>1</sup>

Sangeetha L. Bangolae, Anura P. Jayasumana, V. Chandrasekar  
{sang, anura, chandra}@engr.colostate.edu  
Department of Electrical and Computer Engineering  
Colorado State University  
Fort Collins, CO 80523, USA

## Abstract

*Implementing a TCP-friendly congestion control mechanism is imperative for emerging UDP-based real-time, high-bandwidth applications such as multimedia. VCHILL project for digitized radar data transfer over the Internet is one such application for which a source based TCP-friendly Rate Adaptation Based On Loss (TRABOL) algorithm is deployed and shown to be TCP-friendly over applicable timescales. Comparison of the TRABOL-based radar application with other non-congestion controlled UDP flows shows that the application performs better with TRABOL and is fair towards neighboring TCP flows. Implementation of a simple memory-based mechanism as an add-on to the TRABOL algorithm enhances the performance of the application.*

## 1. Introduction

VCHILL, a Next Generation Internet research project underway at the Colorado State University, USA, aims at a paradigm shift in radar and remote sensing research by exploiting the Internet technology to make digitized radar data available to scientists and researchers in real-time [9]. The main goals of the project include providing interactive, remote access to archived as well as real-time weather radar data and enabling virtual control of the radar operation.

The radar application achieves satisfactory operation with a minimum bandwidth threshold; yet increase in bandwidth provides a better display image at the user end. Tolerance to losses is high, compared to audio and video streaming media. The retransmissions in TCP affect the real-time constraints of the application without providing a commensurate benefit. However, a simple UDP-based implementation would send data at a constant rate, as high as 500 Mbps, regardless of its effects on neighboring traffic, creating network congestion, and resulting in a high loss rate for the radar application itself. Thus a UDP - based digitized radar data transfer protocol (RDP) was designed. As a pure UDP-based implementation will have

a severe negative impact on other TCP streams and in order to make the application safe for deployment in the real Internet, we developed a form of feedback based end-to-end congestion control mechanism, TRABOL for TCP-friendly Rate Adaptation Based On Loss [1,2,3], that is used with RDP.

This paper is an extension of our earlier work [1] in which we propose a TCP-friendly congestion control mechanism for the radar application. A simple memory-based mechanism as an add-on to the deployed congestion control algorithm is shown to increase the performance of the application. The idea behind this mechanism is to adapt to the congestion state of the network based on the recent history of the congestion states instead of simply reacting to the congestion instantaneously. This will help in increasing the overall average throughput of the application while being TCP-friendly [4,6,7,8].

## 2. Memory-based TCP-friendly Rate Adaptation Based On Loss (TRABOL)

According to TRABOL algorithm [1, 3], the server's transmission rate is continually 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 the AIMD algorithm, which has been shown to efficiently converge to fairness [5].

Datagrams are sent to the receiver from the server over the network and the receiver upon receiving the data in a ray sends a feedback message to the sender. ' $L_r$ ' represents the loss rate calculated at the sender based on the feedback received. If ' $L_r$ ' is greater than the maximum loss threshold, MAX\_LOSS, the sending rate is decreased aggressively by D-factor, while if it is less than or equal to the maximum loss threshold and if there was a decrease policy before, an increase policy given by I-factor is

<sup>1</sup> This research was supported in part by DARPA Next Generation Internet (NGI) and NSF Information Technology Research (ITR) Grant no. 0121546.

applied; otherwise the rate is maintained constant [1,2,3]. The memory-based TRABOL can be explained using fig. 1. TRABOL is a source-based rate control mechanism similar in its objective to the popular congestion control mechanism found in TCP; however, in TRABOL, the application-level feedback messages received from the destination about loss rate are used to adjust the transmission rate of the server.

Let  $P_n$  = number of packets sent in a ray of data

$R_t$  = target rate (MAX\_RATE)

$R_m$  = minimum rate (MIN\_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

$L_r$  = ratio of number of packets lost to packets sent

$P_r$  = number of packets received

$n$  = number of packets in a group before a delay

$R_s$  = sending rate.

Due to the coarse granularity of the system clock, we introduce IGG (Inter-Group Gap) between groups of packets to achieve a decrease policy. The number of iterations to remain at a step (after an increase policy is applied), given by 'iteration', is determined by the loss rate and the number of steps, given by 'step\_iteration', is based on the target rate or a loss event occurrence.

With the congestion control algorithm deployed, it was noticed that the transmission rate due to rate control was oscillating between the minimum and maximum rates for every loss event. As the Digitized Radar Signal (DRS) data transfer application generates a long duration flow, it would help to introduce some form of state information about the prior loss events (past history) and current feedback information in the algorithm in order to intelligently adapt to available bandwidth in the network.

According to this new algorithm when a loss occurs in the middle of the congestion cycle, we perceive that the loss must be due to the bandwidth utilization between the previous step and the current step, as seen in Fig. 2. In the next cycle, the rate is increased only until the previous rate where the loss might have occurred. Based on a timer, the maximum rate is probed periodically, so that any available bandwidth may be utilized. Controlling the rate based on the immediate past history and the current feedback information received from the receiver is especially helpful when the bandwidth availability may change during a session, particularly for long-duration sessions such as the radar application.

Analytically, it can be shown that the average throughput of the application with memory-based implementation of TRABOL is greater than that which does not take into account the previous congestion events [3]. Also, it will

enable the application to operate at a stable rate until a loss event triggers the congestion cycle.

```

congestion_manager()
{
  if ( $L_r \leq \text{MAX\_LOSS}$ ) {
    if (set_congestion == 0) {
      make_send_packet(); /*continue to send packets at the same rate*/
       $R_s = P_n/T_t$ ;
      Start TIMER;
      if (TIMER == set)
        Reinitialize step_iteration;
    }
    else if (set_congestion == 1) {
      if (memory_variable != 1) {
        /*Apply the Increase policy */
        /*Increase the rate until TARGET_RATE or loss event occurs*/
        for ( $i=0$ ;  $i < \text{step\_iteration}$ ;  $i++$ )
        {
          /* Increase the rate only if memory variable is not set */
          call delay( $G_a$ ,  $k$ ); /*Decrease the inter-group gap*/
          /*Increase the number of packets being sent by 1*/
          if ( $P_{nn} \neq P_n$ )
             $P_{nn} = P_{nn} + 1$ ;
          for ( $i=0$ ;  $i < \text{iterations}$ ;  $i++$ )
          {
            make_send_packet(); /*Do nothing but maintain the rate*/
             $R_s = P_n/T_t$ ;
          }
        }
        if ( $L_r > \text{MAX\_LOSS}$ ) {
          /* If loss occurs in the middle of a congestion cycle, update the
          states and break */
          Update step_iteration =  $i - 1$ ;
          Store delay and  $P_{nn}$ ;
          /* Set the memory variable to maintain the rate */
          memory_variable = 1;
          break;
        }
      }
    }
    set_congestion = 0;
  }
  else if ( $L_r > \text{MAX\_LOSS}$ ) {
    /*Reduce the rate by D-factor to MIN_RATE*/
    if {
      set_congestion = 1;
       $T_t = P_n/R_t$ ;
       $P_{nn} = R_m * R_t$ ; /*Calculate the new number of packets to be sent */
       $T_a = T_t/P_n$ ;
       $T_r = P_{nn}/R_m$ ;
       $G_t = T_t - T_r$ ; /*Calculate the inter-group gap */
       $G_a = G_t/P_{nn}$ ;
      /*Send the new number of packets with inter-group gap*/
      make_send_packet ();
      call delay( $G_a$ ,  $n$ ); } }
  }
}

```

Figure 1. Memory-based congestion manager

### 3. Performance Evaluation

The performance evaluation plots for the memory-based TRABOL for the radar application are shown below. The test topology and implementation details are detailed in [1]. The server rate in Mbps and the loss rate in % as a function of time for the cases when there is no rate control and when TRABOL is in action at the server end could be referred from [1].

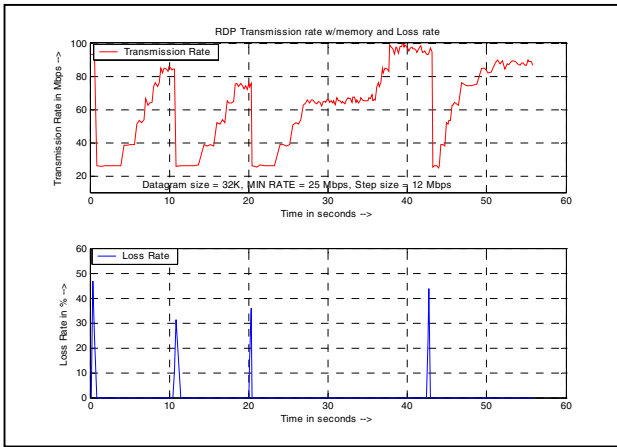


Figure 2. RDP transmission rate and loss rate with Memory-based TRABOL rate control

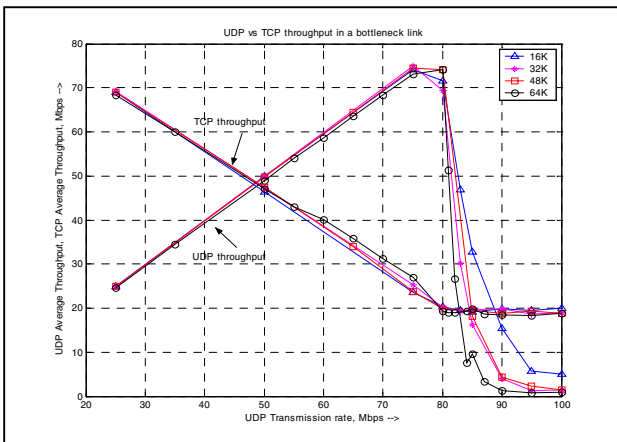


Figure 3. UDP and TCP throughput comparison while sharing a bottleneck link

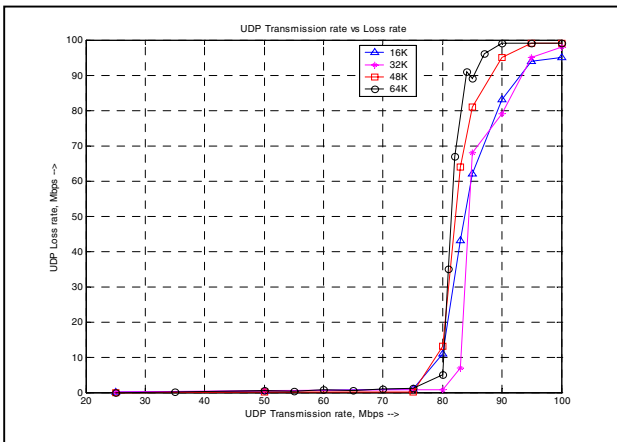


Figure 4. UDP loss rate comparison for various datagram sizes

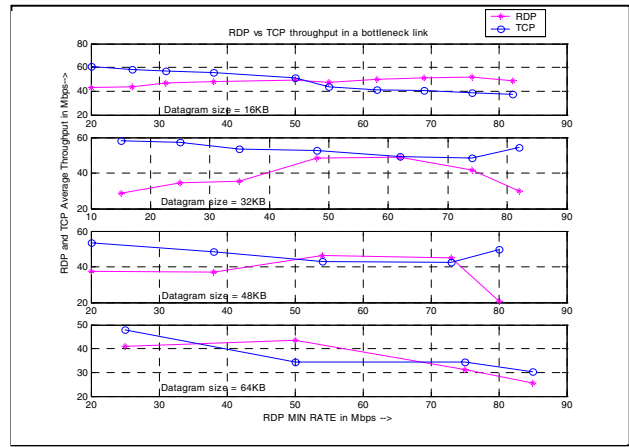


Figure 5. TRABOL-based RDP and TCP average throughput comparison while sharing a bottleneck link

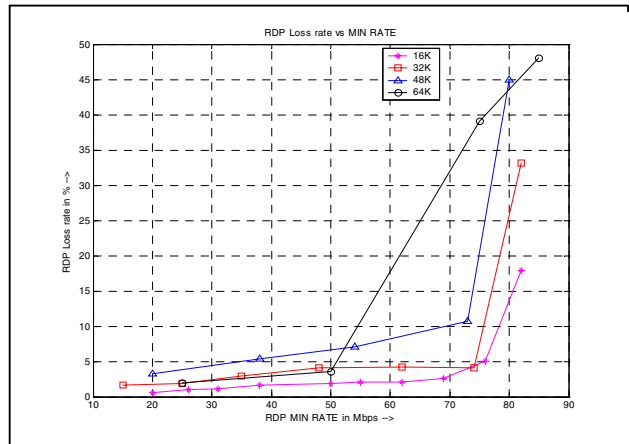


Figure 6. TRABOL-based RDP Loss rate comparison for various datagram sizes

Fig. 2 depicts the performance of the memory-based TRABOL mechanism wherein the server maintains simple state information about the previous loss occurrences and controls the rate adaptively. It can be seen that a loss occurs at about the maximum rate of 90 Mbps and the server remembers this rate when applying the increase policy in the following congestion cycle. The changes in the transmission rates during this period for one complete cycle are shown here for a given datagram size of about 32KB.

Fig. 3 shows the average throughput plots of the *Iperf* based UDP flow and the *netperf* based TCP flow while sharing the same bottleneck link bandwidth of about 100Mbps. In a fair situation, both the flows should receive about 50Mbps each during the duration of co-existence in the bottleneck link. In this case, the transmission rate of the *Iperf* based UDP flow is varied and the corresponding average throughputs over a large timescale are measured. The corresponding loss rates suffered by the UDP-based

flow at the various transmission rates are shown in fig. 4. It can be noticed that the UDP flow continuously transmits at rates greater than 50 Mbps even when there is a neighboring TCP stream, without backing off and thereby consuming an unfair share of the 100 Mbps bottleneck bandwidth. After about 80 Mbps, the link is underutilized showing poor TCP and UDP average throughputs, as the loss rates perceived at the UDP receiver are higher and the TCP-based flow backs off due to its efficient congestion control mechanism while UDP does not.

Fig. 5 gives the details of the TRABOL based RDP co-existing with a TCP flow in a friendly manner for various datagram sizes. The throughput based testing indicates that due to the deployment of the congestion control mechanism in the UDP-based radar application, a neighboring TCP flow gets a fair share of bandwidth. The minimum average bandwidth that TCP received throughout the experiment was only about 33 Mbps as opposed to 18 Mbps when tested in parallel to a non-congestion controlled UDP flow. The loss rates of the RDP protocol with TRABOL for various datagram sizes are shown in fig. 6. The loss rate increases as the datagram size increases and the trend is similar to that in fig. 4, which shows the loss rates suffered by the UDP-based flow at the various transmission rates, but the maximum perceived loss rate for TRABOL based RDP is less than 50%. Throughout the experiments, the MAX\_RATE was set at the link capacity of about 90 Mbps and the MIN\_RATE was increased in steps for study and evaluation.

**Table 1. Comparison of RDP vs UDP**

Minimum average rate (TCP)	UDP	RDP
	18.32 Mbps	34.4 Mbps
Maximum loss rate	99 %	39%

The table above compares the congestion controlled radar data transfer protocol with a non-congestion controlled UDP flow from the measurements made in the previous experiments. The minimum average throughputs that a TCP stream received when sharing the bottleneck link (100 Mbps) with a neighboring UDP and RDP for a particular datagram size and transmission rate settings and the corresponding maximum loss rates suffered by the UDP and RDP flows are also presented here. It shows that while sharing the bottleneck link bandwidth with a TCP flow, RDP performs better than simple non-congestion controlled UDP.

#### 4. Conclusions

This paper provided an overview of the TCP-friendly Rate Adaptation Based on Loss (TRABOL) mechanism implemented for an emerging UDP-based, high-bandwidth application such as digitized radar data transfer over the Next Generation Internet. This is a source based rate control algorithm using the AIMD approach. A simple memory-based mechanism was introduced to adapt to the available bandwidth intelligently. The performance evaluation shows that the memory-based intelligent bandwidth probing mechanism thus increases the average throughput of the application during lightly loaded conditions while being TCP-friendly.

#### 5. References

[1] S. Bangolae, A. Jayasumana, V. Chandrasekar, "TCP-friendly Congestion Control mechanism for a UDP-based High-speed Radar Application and Characterization of fairness", Proc. IEEE-Int. Conf. Communication Systems (ICCS), Nov. 2002, Singapore.

[2] S. Bangolae, A. Jayasumana, V. Chandrasekar, "Gigabit Networking: Digitized radar data transfer and beyond", Proc. IEEE-Int. Conf. on Communication (ICC), May 2003, Alaska, USA.

[3] S. Bangolae, "Design, Implementation and Performance Evaluation study of a TCP-friendly Congestion control Algorithm for a UDP-based Radar Application", M.S Thesis, Colorado State University, Fall 2003 (In preparation)

[4] 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.

[5] 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.

[6] D. Sisalem, H. Schulzrinne, "The Loss Delay Based Adjustment Algorithm: A TCP friendly adaptation scheme", Proceedings of NOSSDAV, Cambridge, UK: July 8-10 1998.

[7] J. Mahdavi and S. Floyd, TCP-Friendly Unicast Rate-Based Flow Control, [www.psc.edu/networking/papers/tcp\\_friendly.html](http://www.psc.edu/networking/papers/tcp_friendly.html): Jan. 1997.

[8] R. Rejaie, M. Handley, D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for real-time streams in Internet", In Proc. of IEEE, INFOCOM 1999: March 1999.

[9] "Virtual CSU-CHILL National Radar Facility", Website: [http://www.engr.colostate.edu/ece/Research/vc\\_hill/vchill.html](http://www.engr.colostate.edu/ece/Research/vc_hill/vchill.html)