Chapter 4

Active node based Congestion Control System for TCP Throughput Enhancement

4.1 Introduction

In the recent past computer networks have experience an explosive growth and with it the congestion problem. In our conventional system, end-to-end connection is established between two endpoints and Routers are assumed to be dummy systems, which forwards packets to the destination. Even though it resides in the network it does not respond to changing state of the network. For e.g., if congestion occurs within network, router will not respond to it, then also it will forwards the packet to the destination thereby increasing network overhead further. Only when endpoint detects congestion, it will reduce its transmission rate. Until then endpoint pushes it’s packet into the network making its condition worse. But if the bandwidth-delay product is high, then it takes long to realize the end point that congestion has occurred.

In this chapter we have attempted to extend the feedback control congestion in to active networks and is applied to TCP congestion control, that is active node based congestion control system (ACC). For implementation of ACC system in NS2 simulator, NS2 is extended wherein the router is made to function as active router by changing the code of NS2. Simulation results show that after incorporating ACC system throughput increases.

4.2 The ACC system description

Active Node Based Congestion Control System (ACC) is a Router assisted dynamic congestion control [92,93] active network based system, which improves the networks performance with large delay-bandwidth product. ACC system can be deployed at various active nodes in the network. ACC system uses active networking technique to enable router participation in both congestion detection and recovery. The feedback congestion control system is extended from the endpoints into the routers. Congestion is detected at the router, which also immediately begins reacting to congestion by changing the traffic that has already entered the network.
Locating both the congestion detection and congestion response at the router removes the feedback delay; the system is stable because changes made at the routers are propagated back to the endpoints. In a conventional feedback system, congestion relief must move from the endpoint to the congestion as the endpoint-sending rate is reduced; in ACC the congestion relief starts at the congested node and the change in state that sustains that relief propagates out to the endpoint.

The TCP implemented at router has the same algorithm as that of standard TCP. The congested router starts traffic modification. When a router detects a packet loss, it calculates the correct window size for the endpoint from information it receives in each packet, and forwards a packet with the new window size to the endpoint.

4.3 Conceptual Working of ACC

ACC includes programs in each data packet that tell routers how to react to congestion without incurring the round trip delay that reduces feedback’s effectiveness in wide area networks [93]. A conventional feedback system will experience a delay in reacting to congestion, such a situation depicted in Figure 4.1. The packet streams from A and B cross at an internal router, C, on the way to endpoint D, congesting C. A feedback system at A or B will detect that congestion, either when it receives notification from the congested router, or when it deduces the existence of congestion because of packet loss or excessive delay. By the time endpoint A has realized C is congested, it has spent at least the propagation delay from A to C and back sending packets as though the network were uncongested, thereby making the congestion worse.

![Fig 4.1 A Congested Network](image)

With ACC, router C has been programmed by the first packet of the connection with instructions on how to react to congestion, and subsequent packets include
information on the current state of the endpoint’s congestion control algorithm. This is depicted in the figure 4.2, which shows that Router C has been programmed using active packets in order to handle congestion.

![Diagram of ACC network reacting to congestion](image)

**Fig 4.2 An ACC network reacting to congestion**

The ACC implementation in TCP, called TCP ACC, follows the same algorithm except that traffic modification begins at router. When a router C detects the packet loss, it calculates the correct window size for the endpoint from information it provides in each packet, and sends a packet with the new window size to the endpoint. Using TCP window adjustment algorithm, the new window size is half the old. The router also installs a filter at the router on the congestion’s path that deletes all packets from that endpoint until it begins to act on the router’s feedback (or it has detected congestion itself). In Figure 4.2 the small circles are the packets being sent to endpoints (A and B) to change their state. The circle in router C represents the active networking component that detects congestion.

### 4.4 Congestion Control in ACC

When C detects congestion, it decides what action the endpoint would take if it had detected congestion in the state reflected by its most recent packet. The router then installs filters that delete packets that the endpoint would not have sent. These filters may be installed at the congested router’s interfaces or at those of neighboring routers (E or F depending on whether congestion was detected by A or B). Finally the congested router sends a message to the endpoint, telling it the new state congestion control system. If this message is lost, the endpoint congestion mechanism will continue to function and deduce a reasonable reaction to congestion; the loss of these
messages is not a catastrophic failure. This process is depicted in Figure 4.2 the small circles are the packets being sent to endpoints (A and B) to change their state. In the wake of those packets, routers E and F being filtering out packets, that have been unsent by the ACC system. The circle in router C represents the active networking component that detected congestion.

Fewer endpoints see congestion at the Router C for two reasons: the congestion is relived sooner, and the ACC system restricts the congestion reaction to fewer endpoints. The congestion is relived because nearby routers reduce its queue length. The nearby routers restrict their traffic editing to packets from the first endpoints that saw congestion. In a conventional system, those first endpoints do not immediately reduce their sending rate, which causes other endpoints to lose packets and reduce their rate. Fewer endpoints reduce their sending rate causes the system as a whole to oscillate less, which improves the global throughput.

![Diagram](image)

**Figure 4.3: Conventional Network when modified with ACC TCP**

Figure 4.3 show how conventional network will look like when ACC is implemented. The same concept is used in simulating ACC.
4.5 ACC System Design

The ACC is implemented in NS2 simulator. NS2 simulator [75-83] was extended to implement the ACC algorithm in the router and at the endpoints. Before implementing the ACC system we represent the system in the form of multiple levels of DFDs to represent increasing flow and functional details of the system.

Based on the details of the ACC system given in the previous section we represent the DFDs.

The DFDs for Active Congestion Control Router could be drawn as follows:

Context Diagram

![Context Diagram](image)

**Figure 4.4 Level-0 DFD Active congestion control**

The Figure 4.4 shows the Context diagram for the Active router. Active router accepts packets from the previous router and depending on the queue conditions it displays the messages, sends the Active feed back packets to the source to slow down the pace or forwards the packets.

**Level -1 DFD**

Level -1 DFD gives more detailed view of the system. Here first router checks whether the queue is full or not. If the queue is full then the packet will be dropped which is indication of the congestion in the system. Then router delivers the packet to be dropped to the congestion recovery module which takes the appropriate actions to relieve the congestion. If the queue is not full then packet is either forwarded or
dropped depending upon the packet source and sequence no. entry in the congestion table, which is explained in detail in the Level -2 DFD.

**Fig 4.5 Level -1 DFD Active Congestion Control System**

**Level -2 DFD**

**Process 1.0 (Congestion Recovery Module-Install Packet Filter)**

Figure 4.6 shows the congestion recovery module in detail. Here first packet is checked for existence in congestion table. If it is first to be dropped from that end-point then congestion table entry of source and dropped packet sequence number is made. After that feed back packet to source (end-point) is sent to reduce its congestion window.

If the congestion table already contains the entry for that end-point then no feed back is sent again and the packet is dropped silently.
Level -2 DFD

Process 2.0 (Drop forward Module)

In the drop forward module the packet source is first checked for the entry in the congestion table. If entry is not found then the packet is simply forwarded. But if entry is found then the packet source and sequence number is compared with the same in the congestion table. If both entry and packet data matches than the entry in the congestion table is removed. Otherwise we drop the out of order packet.
4.5.1 State Transition Diagram for Active Congestion Control Router:

A state is any observable mode of behavior. Each of these states represents a mode of behavior of the system. A state transition diagram indicates how the system moves from state to state. The State Transition Diagrams for Active Congestion Control System is drawn in figure 4.8.
Fig. 4.8 State Transition Diagram Active congestion control system

As shown in the above figure in the system we have three states mentioned below.

- **Idle**
  
  In this state, router waits for the packets to be forwarded by the previous router. As soon as it receives the packet it checks whether congestion is detected or not. Depending on whether the congestion is detected or not it moves to either of the two (Congestion detected or Congestion not detected) states

- **Congestion Detected**
  
  Control reaches in this state if the packet received at the router causes the congestion. In this state router makes entries in the congestion table drops the packet and sends the feedback to the endpoint (source of the packet) to reduce its flow, as explained in the DFD diagrams.

- **Congestion not Detected**

  Control reaches here if the incoming packet does not make queue at the router to drop the packets. Here router either drops the packet or simply forwards it depending on the entry in the congestion table, as explained during the DFD diagrams.

5.4.2 Flow Chart of Active congestion control system

The flow chart for filtering the packets at active router is shown below.
Designing Active TCP

TCP contains a classic, well-understood feedback control system: the congestion avoidance mechanism. Endpoint sending rate is controlled by a sliding window, which is advanced by packet acknowledgements. The size of the window is modulated in response to congestion along the connection’s path.

The window modulations algorithm in TCP is a classic linear increase/multiplicative decrease algorithm. When congestion is detected, the window is reduced to half its current size. When a full window of consecutive packets has been acknowledged without congestion being detected, the window is increased by one maximum-sized packet.

An endpoint deduces that a connection’s path is congested when it detects a packet loss on that connection. Such a loss can be detected by a retransmission timer expiring or by receiving three consecutive acknowledgments of the same packets by the other endpoint.
The ACC implementation is based on TCP, called ACC TCP, follows the same algorithm, expect that traffic modification begins at the congested router. When a router detects a packet loss, it calculates the correct window size for the endpoint from information it provides in each packet, and forwards a packet with the new window size to the endpoint. Using TCP's window adjustment algorithm, the new window is half the old. Then the router installs a filter at the previous router on the connection's path that deletes all packets from that endpoint until it begins to act on the router's feedback (or it has detected congestion itself).

A more sophisticated ACC TCP implementation would install filters that deleted or delayed endpoint packets so that they would appear to the congested router to have been sent by a Slow-Starting endpoint. The current implementation uses the simpler filter that unsends the rest of the current endpoint's window, by discarding packets in flight.

Under ACC, the preferred method of congestion detection is notification by the congested router, but the system still follow the TCP congestion control algorithm in the absence of that feedback. Because TCP only responds to one packet drop per round trip time, packets dropped by the filters will not cause endpoint to close their windows more quickly than they would in the face of a single packet drop.

The reaction to congestion begins at the router with the packet filter installation. This is in contrast to TCP with Explicit congestion Notification (ECN) [16], which uses routers to notify endpoints of congestion, but applies the corrective action from the endpoint. Thus ACC will react more quickly to congestion than TCP with ECN.

**Flow chart for Active TCP**

We have modified the TCP to make it active so that it can interpret the active packets. Algorithm for that is shown below.
As shown in the figure Active TCP will be capable to interpret the active packets. On receiving such packets the TCP will reduce its flow and start re-sending the packets from the sequence number embedded in the Active packets.

We have implemented active router in NS2. In this process we have added following files

congsn-tbl.h This file contains the information regarding maintaining the congestion table in the router.

act-red-tail.h This is the header file for active RED queue. This is almost same as red.h.

act-red.cc This file is an implementation file of the active red queue.
act-drp-tail.h This is the header file for active Drop Tail queue. This is same as droptail.h
act-drp-tail.cc This file contains the implementation details for active drop tail queue.
act-tcp.h This file contains the details of the new TCP header that is added.
act-tcp.cc This file contains the implementation details for making TCP to interpret the active feed back packets from the congested router and start re-sending the packets from the sequence number that is embedded in the active packets.

tcp-sink-actec.h/.cc This file implements the throughput calculation module at the TCP sink end.

Now we have our “physical” structure (files), let’s continue with the “logical” one (classes). The modified class hierarchy of NS2 is as shown in figure 4.11

![Figure 4.11 class hierarchy for ACC system](image)

We have extended the existing TCP Agent so that it can interpret the signals sent by the active router about the congestion and take appropriate steps to relieve the congestion.
4.6 Simulation

In this section the TCP throughput of ACC is compared with that of standard TCP with the aid of simulation. *Network simulator 2* [73-81] is used for simulating ACC and gathering the performance results. Simulations studies of ACC TCP show that it increases endpoint’s average throughput compared to standard TCP in the presence of bursty traffic, and provides comparable performance in a stable network.

4.6.1 Simulation setup

Figure 4.12 shows the topology used for simulations. The bulk traffic source sends data continuously throughout the simulation. All links from an endpoint to a router have a delay of 10ms and bandwidth of 10Mb/s. All endpoints use 1000 byte packets. Each simulation is repeated for different delays on the link from R1 to R2. By varying that delay, the bandwidth delay product that the bulk sources see is changed without changing the bandwidth-delay product that the cross traffic sees.

![Simulation Configuration](image)

**Figure 4.12: Simulation Configuration**

The router R2 is having the Active Congestion Control system, which responds to congestion. Whenever congestion is detected by R2, it finds out the source of congestion and informs the endpoint accordingly. Whenever endpoint receives information of congestion it reduces its transmission rate. R1 and R3 are intermediate non-active router.
We are trying to show that ACC supports high bandwidth-delay product networks. This is shown in simulations by varying the link delay between 10 and 300ms to increase the bandwidth-delay product. This range is selected because it is the common roundtrip time in internet and to demonstrate that ACC is effective across a range of bandwidth delay products.

4.6.2 Simulation case

Simulation is carried out on the topology shown in figure 4.12. Following simulation cases has been tried.

Case I: ACC and TCP comparison using RED router with no cross traffic.

Case II: Simulations using cross traffic, by varying total number of cross traffic Sources

Case III: Simulation using cross traffic by varying Queue length

Case IV: Simulation using cross traffic by varying Sending rate

Case I: Comparison of ACC and TCP using RED router with no cross traffic.

The average outputs of the 10 sources are plotted in the Figure 4.13 where each point is the average throughput of the 10 sources for different delay. The Random Early Detection (RED) system reduces the correlation between burstiness and packet loss, and therefore between slow-starting and packet loss. RED reduces this unfairness by dropping the packet even before queue becomes full. Means this provides feedback even before queue becomes overrun. ACC works transparently across RED gateways, which greatly reduces this intrinsic unfairness. To demonstrate this, we repeated the simulations above using RED queuing at the routers. The results are summarized in Figure 4.13.
**Fig 4.13 RED Router with no Cross Traffic**

**Remarks:** This simulation has demonstrated that TCP ACC gives comparable performance in stable networks. It is observed from Fig. 4.13 that two systems perform nearly identically under stable conditions using RED gateways. RED reduces the unfairness of congestion controls.

**Case II: Comparison of ACC and TCP using RED router with cross traffic by varying total number of cross traffic sources**

These simulations demonstrate that ACC TCP reacts better than endpoint TCP to uncontrolled cross traffic. This simulation includes 10 bulk traffic sources and 7 cross traffic sources. The cross traffic sources have exponentially distributed on and off periods with means of 2.5 seconds and send at 100 Kb/s during their on periods. Cross traffic uses UDP, which does not react to congestion. Each cross traffic source sends for an average of 2.5 seconds at 100 Kb/s and then is quiet for 2.5 seconds. The goal is to model bursty traffic that is not responsive to congestion, for example bursty video source. As before, simulation is repeated for delays ranging from 10 to 300 ms. Here we are varying the total number of cross traffic sources in the simulation and measuring the total throughput of the both ACC system and conventional system.
Figure 4.14 to Figure 4.18 the plots the average throughput seen by the bulk traffic sources against the variable length delay (which directly reflects the change in bandwidth-delay product). Each point is the average of the 10 bulk transfer sources in the simulation. RED queuing is used at routers.

**Fig 4.14: UDP Sources=7**

**Fig 4.15 UDP Sources=10**
Fig 4.16 UDP Sources=15

Fig 4.17 UDP Sources=20
ACC performs better than unmodified TCP. Both systems' performance degrades as the bandwidth-delay product increases, because they do depend on end point router communication, but ACC outperforms TCP. We can see from the above graphs that as we increase the number of the cross traffic sources the ACC performs better at high band-width delay products. The system shows similar performance at small bandwidth-delay products because routers in the ACC system are editing traffic on behalf of the endpoints for only short intervals; the feedback loop is short enough that traditional feedback methods are effective.
Table 4.1: Summary of the results – Average throughput for Varying UDP Sources in KB/sec

<table>
<thead>
<tr>
<th>Varying Delay in ms</th>
<th>UDP Sources=7</th>
<th>UDP Sources=10</th>
<th>UDP Sources=15</th>
<th>UDP Sources=20</th>
<th>UDP Sources=25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Active Router</td>
<td>With Active Router</td>
<td>Without Active Router</td>
<td>With Active Router</td>
<td>Without Active Router</td>
</tr>
<tr>
<td>20</td>
<td>3.70</td>
<td>3.80</td>
<td>3.27</td>
<td>3.22</td>
<td>3.27</td>
</tr>
<tr>
<td>50</td>
<td>3.60</td>
<td>3.55</td>
<td>3.13</td>
<td>3.11</td>
<td>3.13</td>
</tr>
<tr>
<td>100</td>
<td>3.22</td>
<td>3.60</td>
<td>2.95</td>
<td>3.25</td>
<td>2.95</td>
</tr>
<tr>
<td>150</td>
<td>2.78</td>
<td>3.25</td>
<td>2.59</td>
<td>2.67</td>
<td>2.59</td>
</tr>
<tr>
<td>200</td>
<td>3.05</td>
<td>3.22</td>
<td>2.71</td>
<td>2.89</td>
<td>2.71</td>
</tr>
<tr>
<td>250</td>
<td>2.77</td>
<td>2.95</td>
<td>2.64</td>
<td>2.80</td>
<td>2.64</td>
</tr>
<tr>
<td>300</td>
<td>2.35</td>
<td>2.60</td>
<td>2.25</td>
<td>2.50</td>
<td>2.25</td>
</tr>
</tbody>
</table>

Remarks: This simulation demonstrates that TCP ACC reacts better than endpoint TCP to uncontrolled cross traffic. ACC performs better than unmodified TCP. For higher bandwidth delay product ACC gives 18 to 20 percent improvement in throughput.

Case III: Simulation using cross traffic by varying Queue length (buffer capacity)

Here we are varying the queue length at the router in the simulation and measuring the total throughput of the both ACC system and conventional system. Figure 4.19 to Figure 4.21 the plots the average throughput seen by the bulk traffic sources against the variable length delay (which directly reflects the change in bandwidth-delay product). Each point is the average of the 10 bulk transfer sources in the simulation...
(to make figure more legible, throughputs’ are reported in Table 4.2). RED queuing is used at routers.

Fig 4.19 Queue length =25

Fig 4.20 Queue length =30
Table 4.2: Summary of the results – Average throughput for Varying queue length in KB/sec

<table>
<thead>
<tr>
<th>Varying Delay in ms</th>
<th>UDP Sources=7 Queue Length = 25</th>
<th>UDP Sources=7 Queue Length = 30</th>
<th>UDP Sources=7 Queue Length = 35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Active Router</td>
<td>With Active Router</td>
<td>Without Active Router</td>
</tr>
<tr>
<td>20</td>
<td>3.70</td>
<td>3.80</td>
<td>3.76</td>
</tr>
<tr>
<td>50</td>
<td>3.60</td>
<td>3.55</td>
<td>3.52</td>
</tr>
<tr>
<td>100</td>
<td>3.22</td>
<td>3.60</td>
<td>3.22</td>
</tr>
<tr>
<td>150</td>
<td>2.78</td>
<td>3.25</td>
<td>2.80</td>
</tr>
<tr>
<td>200</td>
<td>3.05</td>
<td>3.22</td>
<td>3.09</td>
</tr>
<tr>
<td>250</td>
<td>2.77</td>
<td>2.95</td>
<td>3.08</td>
</tr>
<tr>
<td>300</td>
<td>2.35</td>
<td>2.60</td>
<td>2.65</td>
</tr>
</tbody>
</table>

Case IV: Simulation using cross traffic by varying Sending rate

Here we are varying the sending rate of the UDP sources in the simulation and measuring the total throughput of the both ACC system and conventional system. Figure 4.22 to Figure 4.25 the plots the average throughput seen by the bulk traffic
sources against the variable length delay (which directly reflects the change in bandwidth-delay product). Each point is the average of the 10 bulk transfer sources in the simulation. We are varying the sending rate from 100Kb to 400Kb.

We can see from the figures 4.22 to 4.25 that as we go on increasing the rate of sending more traffic is generated and ACC system again performs better as expected.

Fig 4.22 Rate=100Kb

Fig 4.23 Rate=200Kb
Histogram for the simulation

Histogram is used to divide the data in the range. Here we use the histogram to indicate that in ACC we have more number of the endpoints with the higher
throughput, because as the endpoints execute the Slow-Start algorithm, they are sending more packets back to back. These back to back packets are more likely to be lost, which perpetuates the cycle. The result is that some endpoints experience very few losses, and some experience many.

Histograms for both systems are shown in the figure 4.26 and figure 4.27 for the delay 150ms. We can see from the figure that with active router we have more number of end-points with higher throughput which increases the average throughput of the system compared to conventional system without active router.

Histogram for ACC v/s Conventional System

![Histogram](image)

**Fig 4.26 Histogram with no Active Router**
4.7 Conclusion

This chapter has focused on describing how Active Networking can be used to augment feedback congestion control and that there are tangible benefits to doing so. For implementing Active node based Congestion Control (ACC) system in ns2, ns2 capabilities are extended by adding modified new TCP agent ACC TCP to function router as active router. Active queues are also added. By making the router as active router, early packet is processed at router thereby improving the throughput.

A simulation result shows that when Active Congestion Control System is incorporated, the throughput under bursty traffic conditions improves about 20 percent than standard TCP.

With active router we have more number of end-points with higher throughput which increases the average throughput of the system compared to conventional system without active router.

We assert through simulation that with varying buffer size ACC system performs better than TCP.