Chapter 3

3. Multicast Support for Effective Cache Consistency in Mobile Environment

The increased demand for wireless technology and related applications has seen the introduction of a wide range of wireless products such as laptops, cellular phone, etc., to meet customers’ demand while preserving high efficiency and data integrity that suit the needs of a broad range of Mobile Users (MUs). Ideally, an MU should be able to have access to desired information such as news, financial information, stock prices, etc. whenever and wherever he desires. However, the mobile environment is different and faces two major limitations, namely the system has limited bandwidth, and the mobile user is restricted by limited power device. Hence, caching can become a viable technique to support effective service by having a local copy of the data at the user's terminal. However, if there are changes with the original copy of the data, the local copy at MU will not be valid any more requiring the MU to verify the validity of the data to achieve consistency before responding to a query. Other features of the mobile environment that could also impede achieving high consistency is the frequent
disconnections of MU which makes data communication in such networks much more challenging than in wired networks. The characteristics of the mobile devices also need to be taken into account when designing applications for such environments.

### 3.1 Mobile Device characteristics

**Battery**

Due to the limited battery power of mobile devices, applications need to be designed as energy-efficient as possible to avoid the processes from execution when they are idle. This saves the CPU usage to some extent.

**Connections**

The applications can take the benefit of wireless connections, but the delay or drop of the connections especially for synchronous processes should also be considered.

**Execution of applications**

If applications are interrupted due to the receipt of SMS or a phone call, in this situation a timely and smooth recovery of the application is needed to be ensured.

**Memory**

The mobile environments are constrained by storage space (dynamic memory-runtime heap memory), therefore data structures for efficient memory utilization must be used. The flexibility of reuse of code must also be provided.
**Processing power**

The CPUs in some mobile devices are not as effective and powerful as those that are used in Personal Computers (in desktop environment), therefore the applications should be developed according to different set of specifications.

**Usability**

The features of graphical user interface such as size of screen (small) may have great impact on the usability of applications on a mobile device. These features have to be considered at the time of development.

**User interface**

The applications are developed for small screens, so the input methods differ from those that are used in desktop environment. Also the types of devices pose a set of constraints due to variety of displays, keyboards that offer different look and feel for mobile devices.

The limitations mentioned above restrict maintenance of the local copy or replica, a very important issue in mobile environments as the mobile users require local copies when they are disconnected from the network otherwise they have to use inconsistent data in a degraded network connection. Hence in wireless mobile communication environments data caching is crucial as it reduces contention by increasing the probability of desired data for the nodes, enhancing the system performance. The important issue that needs consideration is
how the data consistency between the client cache and the server can be maintained.

Caching of most commonly accessed data items reduces the access delay associated with answering a query contributing towards saving the bandwidth and improving overall system performance. The roaming of MU and the repeated disconnections complicates the cache consistency maintenance mobile computing environments. A successful strategy hence needs to efficiently handle both disconnectedness and mobility.

The existing types of cache consistency maintenance methods for mobile computing environments - stateless and state-ful are discussed in Chapter 2.

In a stateless approach, the server has no information about the contents cached by mobile user (MU). Usually the invalidation report (IR) is broadcasted periodically to all the MUs. The Stateless approaches are less powerful and even offer weak support under disconnectedness and less scalable.

Different stateless algorithms proposed include Timestamps (TS), Amnesic Terminals (AT), and Signature (SIG), in which the BS broadcasts Invalidation Report (IR messages) every ‘t’ seconds. An IR message includes all data item-IDs updated during the past ‘kt’ seconds, where k is a positive integer. The advantage of these algorithms is that a BS does not maintain any state information about its MUCs (Mobile User Cache),
thus allowing simple management of the BSC (Base Station Cache). However they suffer from the following drawbacks:

1. They cannot manage fast updating data systems on account of increased number of IR messages. They are not scalable.

2. The average-access-latency is always longer than half of the broadcast-period because of all requests getting answered only after the next IR.

3. All cache entries are deleted, if the sleep-time (time during which an MU is disconnected from its BS) is longer than ‘kt’. This leads to unnecessary bandwidth consumption, particularly if the data item is valid.

To handle the long sleep-wakeup patterns, several algorithms have been proposed. For example - Bit-sequence (BS) algorithm in which data entries in the cache will be invalidated, only if all or more than half of cache entries get deleted. However, this model requires the broadcast of a larger number of IR messages than TS and AT schemes. The uplink validation check scheme deals with long sleep-wakeup patterns but it requires more uplink bandwidth. Finally fails to handle very long sleep-wakeup patterns. In order to reduce the IR messages, adaptive methods have been developed. For implementing broadcast, different IR-strategies based on update-frequency, MU access and sleep-wakeup patterns have come into existence. Further an absolute validity interval (AVI) is associated with every data item but this strategy also fails to reduce the
consequences of delays introduced due to periodic broadcast. The
approaches discussed above can only benefit the MUs only if the data
items retrieved are same within a broadcast cycle. On the other hand if
the MUs retrieve similar data in separate broadcast cycles, then they
cannot share the broadcast data. The broadcast in this scenario becomes
inefficient and also sensitive to the number of MUs within the cell.

Existing caching schemes are based on an assumption that there
is absolute connectivity thus reliable communication between BS and
MUs for IR broadcast. Reliable communication mechanism means
acknowledged data communication. But soon after an IR is broadcast
due to the rise in competition for uplink between the BS and MUs there
will be an impact on the uplink queries i.e. the average access delay and
MU’s battery consumption also rises. Also if there is IR broadcast and at
this instant an MU is disconnected, the server does not receive any
acknowledgement and it retransmits the IR thinking that the IR is lost or
the MU is disconnected. Moreover, the existing schemes proposed in the
literature do not study the possible inconsistency and performance loss
due to wireless channel errors. Thus, there is a need for scalable and
efficient algorithms for maintaining cache consistency in the error-prone
wireless channels.

Broadcast has the advantage of being able to serve an arbitrary
number of MUs with minimum bandwidth consumption. Thus, efficient
mobile data transmission architecture must carefully design its
broadcast and cache management schemes to maximize bandwidth utilization and also to minimize average query delay. Additionally, such architecture should be scalable to support large database systems as well as a large number of MUs. The various cache consistency maintenance algorithms proposed for wireless mobile environments are studied in detail in chapter 2. They include an asynchronous stateful (AS) algorithm in which a BS records all retrieved data items for each MU. When an MU first retrieves a data-item after it wakes up, (based on the MUC content record and sleep-wakeup time) the BS sends an IR to that particular MU. Whenever a BS receives an update from the server for each recorded data-item, it immediately broadcasts that item’s IR to MU.

Fig 3.1 Mobile Communication Environment
The advantage of the AS scheme is that the BS avoids unnecessary IR broadcast to MUs. Also MUs deal with any sleep-wake up pattern and do not lose valid data-items. BS must record all cached data-items for each MU in order to maintain each MUC. In this situation an MU is restricted to download data-items which it requested through the uplink, thus making the broadcast utility less efficient and even terribly sensitive to the number of MUs. A counter-based scheme has been proposed in literature that is more suitable. It saves unnecessary IR traffic by identifying hot data. The MU must piggyback the data transformation to the server, whenever the contents of MUC gets altered - thus consuming battery power and uplink bandwidth. The above schemes also assume reliable data transfer. The cost of reliable data transfer for broadcast (or multicast) is significant because of asymmetric nature of the uplink and downlink data transmission. It is necessary to allow a maximum number of retransmissions of the acknowledgment of IR broadcast (or multicast) because of this channel competition.

The two fundamental information delivery methods for wireless data applications are discussed below: point-to-point access and broadcast.

Point-to-point access: There exists a logical channel between the client and the server to submit queries to the server and get the results returned as in a wired network.
Broadcast: In this method the data is sent to all. The client can select the desired data. Further in on-demand broadcast, the client submit queries to the server so that the desired data are guaranteed to be broadcast. Broadcast is an attractive method for various reasons:

- In a single broadcast all the outstanding requests corresponding to a data-item are serviced simultaneously. As such, broadcast can accommodate a huge number of users.
- In mobile wireless communication environments, the downlink capacity is much greater than the uplink capacity.
- In a mobile wireless communication system, broadcast do not incur additional cost since it is inherently employed to deliver information. The point-to-point and broadcast systems share many issues like improving response time by energy conservation and less bandwidth consumption, this chapter focuses on broadcast systems only. Energy conservation (means the client should exclusively be made to access what it desires the most) and efficiency of data access (means how fast a request is serviced) are very vital to any wireless data system. The limited energy/power on mobile clients range between a few hours to less than a day under rigorous use and only battery capacity of 20–30% is anticipated in future [31]. In the literature access-time and tune-in time are the parameters of concern for broadcast to measure efficiency of access and energy conservation. A broadcast schedule is carefully designed determining when and what should be broadcasted by the
server. The three kinds of broadcast models: Push-based, on-demand (or pull-based) and hybrid broadcast are studied in detail in chapter 2. Consequently, there are three kinds of data scheduling methods push-based, on-demand and hybrid scheduling corresponding to these data broadcast models. In broadcast model a data item can be retrieved by a mobile user only when it monitors the channel continuously. This will consume appreciable amount of battery power. Therefore air indexing technique is used.

In view of the problems cited above when broadcast based algorithms are used for maintenance of cache consistency in mobile environment we take into consideration the concepts of existing multicast approaches of routing packets to wireless, mobile adhoc networks viz. MAODV, ODRMP and multi-rate protocol like RLM and propose an MCAODV which is an enhanced-MAODV multicast approach for maintaining cache consistency in mobile environment.

3.2 MCAODV – An Enhanced multicast flow control protocol

3.2.1 Description

In this section we explain and experiment how to support multicasting in wireless networks without affecting the dominant unicast flows. Protocols like TCP/IP are used for the congestion control[23] when unicast flows are used during data transmission. But multicast flows are not supported by any such flow control mechanism in wireless networks.
We propose a flow control scheme for preventing severe congestion when multicast flows are used. This scheme is based on the two concepts (1) the layered multicast concept and (2) the idea of routing-based congestion avoidance. Multicast On-demand Distance Vector (MAODV) Routing Protocol and On-demand Distance Multicast Routing Protocol (ODMRP) [16, 18] the commonly used routing protocols for multicasting in wireless networks. These protocols set up routing information in nodes similar to unicast protocols but they do not have any other control over the flows, such as congestion control.

We propose a localized scheme in wireless networks at the network layer level to support multicasting while still maintaining fairness in terms of unicast flow. Every independent node acts on information collected locally without any additional information about the interaction between the nodes. In the given multicast strategy the multicast source encodes its signal into several layers of various priorities and sends each layer to a separate multicast group. Receivers of the multicast source subscribe to these multicast groups and packets for all or some of these groups flow into the receivers. At the same time, each intermediate node in the wireless network monitors its wireless link. Whenever the link is found to be congested, the node cuts the rate of multicast flows so far aggregated if bandwidth utilization is in excess used by the multicast flows on the link.
3.2.2 Support for Flow Control Operation - Retrieving Flow Information

First step in proposed scheme is a node collects flow information about the traffic traversing its link to support the operation of congestion control and retrieving the information about the number of TCP flows, number of multicast flows, number of layers of each multicast flow, average per-flow rate of TCP flows and the average per-flow rate of multicast flows. The source and destination addresses and port numbers are used to identify the TCP flow or the layer of a specific multicast flow to which a packet belongs. The number of layers that the multicast flow has on a link is obtained by observing the number of different multicast addresses used by the packets of the flow traversing the link. For the average per-flow rates, the proposed scheme does not need the absolute values. Instead, the average per-flow rates of TCP flows and of multicast flows are measured in the following way. The total number of TCP packets and the total number of multicast packets traversing the link in specified intervals are counted, and then divided, respectively, by the number of TCP flows and the number of multicast flows traversing the link. The results are the measured average per-flow rates of TCP flows and multicast flows. In addition, the counting process is reset and restarted whenever congestion occurs at a bottleneck. The flow information retrieved from the traffic traversing a link is the basis for the operation of the proposed scheme on the link. The proposed
scheme assumes that the number of layers that a multicast flow possesses on a link can reflect its relative data rate among the multicast flows traversing the same link. Therefore on the same link, a multicast flow with more layers has a higher data rate than a multicast flow with fewer layers.

**Channel Assignment**

A multicast flow is defined as a link originating from a multicast source and transferring all the packets to a respective group and the source node in the proposed scheme creates layers to this multicast group. When the packet transfer occurs the wireless link of the node monitors the output queue at regular intervals. If the number of packets exceeds a specified threshold the packets that follow are blocked and allocated to other layers for transmission.

The other layers are also monitored for a normal flow. Wherever there is congestion, the control operation changes the average number of flows and allocates the flows depending upon the previous calculations in future. The source node address and destination node address are used to calculate and identify the TCP flows and the layer information for multicasting. The methods used in our system begin by retrieving the flow information based up on the average values of the flow calculated and then these are compared with TCP flows.

In the proposed scheme, the functions of the receiver are enhanced so that the receiver waits for certain interval of time before adding a new
layer. To select the proper channel or layer – (i) the channel sum is calculated for each link in the routing table depending on the bandwidth of channels (ii) then the maximum sum of channels is calculated. The above procedure is repeated on every node having the routing tables. This sum which is calculated (for each channel assignment) is called to resolve the value the routing table for the selection of the best channel from the routing table and then onwards it is used to take the routing decisions. This helps in checking whether adding a new layer blocks the network or not, if it doesn't then the layer can be added to the receiver and the normal flow of packets is realized at the receiver without disrupting other packet flows.

3.2.3 Scenario Creation and Scheme Evaluation

This section explains the creation of scenario for the proposed scheme. The evaluation of the proposed scheme was done by creating scenario in Network Simulator 2.34 (NS-2.34) and performing the simulations by varying different parameters. The values of parameters used in simulations are explained in implementation section. The simulation configuration created a single typical bottleneck in a wireless network that bears both multicast and unicast flows and additionally adopts a fully localized strategy for controlling multicast flows. The congestion within a network is resolved locally and we do not stress on the cooperation between intermediate nodes.
The wireless network configured in the simulator is based on a two-ray ground propagation model. The MAC protocol is IEEE 802.11, the ad hoc routing protocol is MCAODV and the link queue size is 50 packets. The nodes are placed in a network area of 670 m X 670 m with a node transmission distance of 100 m. Two nodes at a distance of 100 m form a shared wireless bottleneck. The senders of the competing flows are randomly placed within 100 m around one node of the shared bottleneck and the receivers of the competing flows are randomly placed within 100 m around the other node of the shared bottleneck. It is ensured though that all the senders must be more than 100 m away from all the receivers so that all competing flows traverse the shared bottleneck, which has a radio bandwidth of 1 Mbps. There are five competing flows in our simulations, two multicast flows, and three TCP flows, and there is a maximum of 14 nodes in the simulations. Each multicast flow has 15 layers and the size of each layer is 10 Kbps.

In addition, a multicast source uses either a Constant Bit Rate (CBR) source or Variable Bit Rate (VBR) source. It is easier to deal with CBR multicast sources. The VBR multicast sources introduce traffic fluctuation on links. Therefore VBR multicast sources are more challenging for existing multicast congestion control schemes. We use two scenarios to test the proposed scheme for each traffic source. In the first scenario, there is no node mobility in the second scenario nodes follow random waypoint movement.
**Implementation**

This section of the chapter explains the design considerations and several design diagrams such as use case diagram, class diagram, and sequence diagrams. We also describe the important coding techniques which are being used. The complete code is not included, only the sections which are most important and critical are included in this section of the chapter.

### 3.2.4 UML Diagrams

![Use Case Diagram](image)

**Fig 3.2 Use Case Diagram**
Fig 3.3 Sequence Diagrams (Random Topology)

Fig 3.4 Sequence Diagrams (Random Traffic)
Fig 3.5 Sequence Diagrams (Congestion control)

Fig 3.6 Sequence Diagrams (Analysis)
Mobile node placement, Number and X, Y, Z coordinate ranges

13 mobile nodes, range values of X: 215 - 430  Y: 190 - 300  Z:0 - 0

Traffic Pattern

In this module we implement multi flows through the use of constant bit rate (CBR) and variable bit rate (VBR) traffic depending upon simulation. We create different flows by attaching TCP created agents on source and sink which are source and destination respectively. In TCP traffic we create flows between individual source and destinations whereas in Multicast traffic (MCT) flows we use two different sources connecting to individual multicast flows. Wherever the multicast flows are computed they are assigned to send packet starting at 100 second.

**Setup multicast traffic flow between nodes**

```plaintext
set tcp1 [new Agent/TCP]
```
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $tcp1
$ns_ attach-agent $node_(11) $sink1
$ns_ connect $tcp1 $sink1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set random_ 1
$cbr1 attach-agent $tcp1
$ns_ at 100.0 "$cbr1 start"

set tcp2 [new Agent/TCP]
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $tcp2
$ns_ attach-agent $node_(8) $sink2
$ns_ connect $tcp2 $sink2

set cbr2 [new Application/Traffic/CBR]
$cbr2 set random_ 1
$cbr2 attach-agent $tcp2
$ns_ at 100.0 "$cbr2 start"

set tcp3 [new Agent/TCP]
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $tcp3
$ns_ attach-agent $node_(7) $sink3
$ns_ connect $tcp3 $sink3

set cbr3 [new Application/Traffic/CBR]
$cbr3$ set random_ 1
$cbr3$ attach-agent $tcp3$
$ns_ at 100.0 "$cbr3$ start"

**Setup unicast traffic flow between nodes**

set tcp1 [new Agent/TCP]
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $tcp1
$ns_ attach-agent $node_(11) $sink1
$ns_ connect $tcp1 $sink1

set cbr1 [new Application/Traffic/CBR]
$cbr1$ set random_ 1
$cbr1$ attach-agent $tcp1
$ns_ at 100.0 "$cbr1 start"

set tcp2 [new Agent/TCP]
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(4) $tcp2
$ns_ attach-agent $node_(8) $sink2
$ns_ connect $tcp2 $sink2

set cbr2 [new Application/Traffic/CBR]
$cbr2$ set random_ 1
$cbr2$ attach-agent $tcp2
$ns_ at 100.0 "$cbr2 start"

set tcp3 [new Agent/TCP]
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(5) $tcp3
$ns_ attach-agent $node_(12) $sink3
$ns_ connect $tcp3 $sink3
set cbr3 [new Application/Traffic/CBR]
$cbr3 set random_ 1
$cbr3 attach-agent $tcp3
$ns_ at 100.0 "$cbr3 start"

**Compute Throughput**

In this section we create a method in the TCL file to record the number of bytes received at each sink. Using this compute the expression for calculating the combined throughput for multicast flows (MCT) on three links. We set a time interval at 1 (millisecond) and write compute flows to the output file. Reinitialize the receive to zero then record again. A function is used to compute the number of bytes in the next second.

**Pseudo code for calculating the Throughput for Multicast Flows**

```
proc record {  
    #Declare the variables sink1 sink2 sink3 sink4 sink5 sink6 f0 f1  
    #Set a Simulator instance  
    #Set the time at 1.0 ms  
    #How many bytes have been received by the traffic sinks?  
    # Set the bandwidth variables
```
set bw0 [sink1 set bytes]
set bw1 [sink2 set bytes]
set bw2 [sink3 set bytes]
set bw3 [sink4 set bytes]
set bw4 [sink5 set bytes]
set bw5 [sink6 set bytes]

# Get the current time
set now [ns now]

# Calculate the throughput (in Kbytes/s) and write it to the files
puts $f0 "$now [expr ($bw0+$bw1+$bw2)/$time*8/10000/3]"
puts $f1 "$now [expr ($bw3+$bw4+$bw5)/$time*8/10000/3]"

# Reset the bytes_ values on the traffic sinks
$sink1 set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0
$sink4 set bytes_ 0
$sink5 set bytes_ 0
$sink6 set bytes_ 0

# Re-schedule the procedure
$ns at [expr $now+$time] "record"
}
3.2.5 Formulation of Channel Assignment Function

To select proper channels, the channel sum is calculated for each link in the routing table. Then the maximum sum is calculated for each node which has routing table. This sum which is calculated will be assigned to each channel. The value of the sum will be called to resolve the routing table which chooses the best channel from the routing table and this assigned one is used to make the routing decisions and selecting the channel.

//ChannelMCAODVCal is used to calculate the new routing metric MCAODV

double MCAODV::ChannelMCAODVCal(mcaodv_rt_entry *rt)
{
    #Declare the variables for finding the sum of channels on the links and initialize it to zero ( MCAODVSum = 0 )
    #Declare an array for maximum channels available for the multicast flows ( ChannelMCAODVSum [ 15 ] )
    for ( int i = 0 ; i < rt->linkstateindex ; i++ )
    {
        MCAODVSum = MCAODVSum + rt->rt_linkstate[i].linkMCAODV ;
        #Find out the MCAODV sum on each channel
        switch (rt->rt_linkstate[i].currentchannel)
        {
        }
case 0:
   ChannelMCAODVSum[0] = 
   ChannelMCAODVSum[0] + rt -> rt_linkstate[i].linkMCAODV ;
   break;

case 1:
   ChannelMCAODVSum[1] = 
   ChannelMCAODVSum[1] + rt -> rt_linkstate[i].linkMCAODV ;
   break;

case 2:
   ChannelMCAODVSum[2] = 
   ChannelMCAODVSum[2] + rt -> rt_linkstate[i].linkMCAODV ;
   break;

case 3:
   ChannelMCAODVSum[3] = 
   ChannelMCAODVSum[3] + rt -> rt_linkstate[i].linkMCAODV ;
   break;

case 4:
   ChannelMCAODVSum[4] = 
   ChannelMCAODVSum[4] + rt -> rt_linkstate[i].linkMCAODV ;
   break;

case 5:
   ChannelMCAODVSum[5] = 
   ChannelMCAODVSum[5] + rt-> rt_linkstate[i].linkMCAODV ;
break;

case 6:
    ChannelMCAODVSum [ 6 ] =
    ChannelMCAODVSum [ 6 ] + rt -> rt_linkstate[i].linkMCAODV;
    break;

case 7:
    ChannelMCAODVSum [ 7 ] =
    ChannelMCAODVSum [ 7 ] + rt-> rt_linkstate[i].linkMCAODV;
    break;

case 8:
    ChannelMCAODVSum [ 8 ] =
    ChannelMCAODVSum [ 8 ] + rt-> rt_linkstate[i].linkMCAODV;
    break;

case 9:
    ChannelMCAODVSum [ 9 ] =
    ChannelMCAODVSum [ 9 ] + rt -> rt_linkstate[i].linkMCAODV;
    break;

case 10:
    ChannelMCAODVSum [ 10 ] =
    ChannelMCAODVSum [ 10 ] + rt -> rt_linkstate[i].linkMCAODV;
    break;

case 11:
    ChannelMCAODVSum[ 11 ] =
break;

case 12:
    ChannelMCAODVSum[12] = 
    ChannelMCAODVSum[12] + rt -> rt_linkstate[i].linkMCAODV;
break;

case 13:
    ChannelMCAODVSum[13] = 
    ChannelMCAODVSum[13] + rt -> rt_linkstate[i].linkMCAODV;
break;

case 14:
    ChannelMCAODVSum[14] = 
    ChannelMCAODVSum[14] + rt -> rt_linkstate[i].linkMCAODV;
break;

default:
    ; //default do nothing
} //switch

int maxChannelMCAODVSum = 0;

//Find out the max ChannelMCAODVSum on channel j
for (int i=0; i<15; i++)
{

if ( ChannelMCAODVSum[ i ] > maxChannelMCAODVSum )
{
    maxChannelMCAODVSum = ChannelMCAODVSum [ i ] ;
}

return ( (1-mcaodvbeta) * MCAODVSum + mcaodvbeta * 
         maxChannelMCAODVSum ) ;


3.3 Experiments and Results

3.3.1 Design of test cases and scenarios

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case Name</td>
<td>Required NS2 Software Testing</td>
</tr>
<tr>
<td>Purpose</td>
<td>To check weather the required NS 2 Software is installed on the systems</td>
</tr>
<tr>
<td>Input</td>
<td>Enter NS on command terminal</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Should Display the % symbol</td>
</tr>
<tr>
<td>Actual Result</td>
<td>Displays % symbol</td>
</tr>
<tr>
<td>Result</td>
<td>Success</td>
</tr>
</tbody>
</table>

Table 3.1 NS2 Software Testing
<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case Name</td>
<td>Required NAM Software Testing</td>
</tr>
<tr>
<td>Purpose</td>
<td>To check whether the required NAM Software is installed on the systems</td>
</tr>
<tr>
<td>Input:</td>
<td>Enter NAM on command terminal</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Should Display the Network Animator</td>
</tr>
<tr>
<td>Actual Result</td>
<td>Displays Network Animator</td>
</tr>
<tr>
<td>Result</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Table 3.2 NAM Software Testing**

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case Name</td>
<td>Programs Integration Testing</td>
</tr>
<tr>
<td>Purpose</td>
<td>To ensure that all the modules work together</td>
</tr>
<tr>
<td>Input</td>
<td>Open and Run the Project</td>
</tr>
<tr>
<td>Expected Result</td>
<td>All the code should run in the common execution Environment</td>
</tr>
<tr>
<td>Actual Result</td>
<td>All the code Executed</td>
</tr>
<tr>
<td>Result</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Table 3.3 Programs Integration Testing**
<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case Name</td>
<td>Scenario File Generation testing</td>
</tr>
<tr>
<td>Purpose</td>
<td>To ensure that Scenario file is generated for the required Nodes, Pause Time and Connection Source</td>
</tr>
<tr>
<td>Input</td>
<td>Input number of nodes and pause time</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Should generate the Scenario File</td>
</tr>
<tr>
<td>Actual Result</td>
<td>Scenario file is generated</td>
</tr>
<tr>
<td>Result</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Table 3.4** Scenario File Generation testing

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case Name</td>
<td>Connection Pattern Generation testing</td>
</tr>
<tr>
<td>Purpose</td>
<td>To ensure that Connection Pattern file is generated for the required Nodes, Pause Time and Connection Source</td>
</tr>
<tr>
<td>Input</td>
<td>Input number of nodes and Simulation time and Number of Source</td>
</tr>
<tr>
<td>Expected Result</td>
<td>Should generate the Connection Pattern</td>
</tr>
<tr>
<td>Actual Result</td>
<td>Connection Pattern file is generated</td>
</tr>
<tr>
<td>Result</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Table 3.5** Connection Pattern Generation testing
3.3.2 Constant Bit Rate (CBR) Multicast Sources

This section provides the description of the simulation results for constant bit rate (CBR) multicast sources. We used two scenarios to test the proposed scheme. In the first scenario, there is no node mobility, in the second scenario, nodes follow random waypoint movement, with five competing flows, two multicast flows and three TCP flows.

Static Network Scenario

![Graphs showing individual multicast throughput and number of layers in static (CBR).](image)

**Fig 3.8 Individual multicast throughput and number of layers in static (CBR).**

In static constant bit rate (CBR) scenario there is no node movement, all flows start at the 100th second of the simulation (for
leaving some establishment time for the wireless network) and stop at the 1,500th second.

The figure 3.8 shows the throughput of two multicast flow and number of layers for each multicast flow which is used to reducing the congestion. As shown in these figure both multicast flows gets a throughput close to 5 Kbytes/s. Moreover, after the initial adjustment, the number of layers of each multicast flow is fairly stable even though there is one or two layers of difference between the two multicast flows in some cases. One or two layers of difference between multicast flows is possible with layered multicast congestion control because the units of rate adjustment for multicast flows are layers.

In this figure, we can find that the proposed scheme achieves high stability in both numbers of layers and throughput for both multicast sessions. This demonstrates that the congestion is controlled very well with the proposed scheme.
Fig 3.9 Individual TCP throughput and average per-flow TCP throughput in static (CBR).

In static constant bit rate (CBR) scenario there is no node movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The figure 3.9 shows the throughput of each individual TCP flow and the average per-flow throughput of TCP flows, which specified the throughput at different levels. As shown in these figure, each unicast flow, gets a throughput that is close between 4 to 5 Kbytes/s.

From this figure, we can find that the proposed scheme achieves high stability and fairness in throughput for all TCP sessions. This
demonstrates that the congestion is very well controlled with the proposed scheme.

**Random Node Movement Scenario**

![Graphs showing individual multicast throughput and number of layers in random (CBR).](image)

**Fig 3.10 Individual multicast throughput and number of layers in random (CBR).**

In random constant bit rate (CBR) scenario all the nodes are moving it means that both the source and destination have a movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The Fig 3.10 shows the throughput of two multicast flow and number of layers for each multicast flow which is used to reducing the
congestion, hence in first multicast flow (MCT1) when we are start
sending packets at 100 seconds we see the throughput till 258 second
after that there is no throughput till 750 second because since the nodes
are moving it may not be in the range with any other nodes at that time
and we have specifically defined the node range to be within 100 meters
to communicate. Therefore even the number of layers is not used during
this period of time because the layers are created depending up on the
throughputs in the first multicast flow (MCT1), with respect to second
multicast flow (MCT2) since there are two flows considered with different
sources and one source may be moving away but other source still in the
range, therefore there is a throughput which is available in second
multicast flow even the number of layers are available.

As shown in figure, the two multicast flows do not have a stable
number of layers anymore as random movement is introduced to nodes
in the network. This is because when nodes move, link quality and
actual capacities change in the network.
Fig 3.11 Individual TCP throughput and average per-flow TCP throughput in random (CBR).

In random constant bit rate (CBR) scenario all the nodes are moving it means that both the source and destination have a movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The Fig 3.11 shows the throughput of each individual TCP flow and the average per-flow throughput of TCP flows, which specified the throughput at different levels, In first TCP flow (TCP1) the throughput in constant range till 1100 second after that the flow varies since the source
node may be moving away to out of range because all nodes have a movement in random scenario.

But where as in second and third graph which shows the throughput of second TCP flow (TCP2) and third TCP flow (TCP3) there is closeness in constant throughput, even in some time the throughput gets changed, this is because of node movement i.e. the nodes move, link quality and actual capacities change in the network, the TCP_Average graph shows the average throughput which is calculated for TCP1, TCP2 and TCP3.

### 3.3.3 Variable Bit Rate (VBR) Multicast Sources

This section shows how the proposed scheme behaves with variable bit rate (VBR) multicast sources. For generating the Variable Bit Rate (VBR) multicast sources, we used the exponential traffic model in NS-2. The simulated variable bit rate (VBR) traffic has exponentially distributed burst and idle times. We used two scenarios to test the proposed scheme. In the **first scenario, there is no node mobility, in the second** scenario, nodes follow random waypoint movement, with five competing flows, two multicast flows and three TCP flows.
Static Network Scenario

**Fig 3.12 Individual multicast throughput and number of layers in static (VBR).**

In static variable bit rate (VBR) scenario there is no node movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The above figures in Fig. 3.12 shows the throughput of two multicast flow and number of layers for each multicast flow which is used to reduce the congestion. As shown in these figures both multicast flows gets a throughput close to 5 Kbytes/s. Moreover, after the initial
adjustment, the number of layers of each multicast flow is fairly stable, even there is one or two layers of difference between the two multicast flows in some cases. One or two layers of difference between multicast flows is possible with layered multicast congestion control because the units of rate adjustment for multicast flows are layers.

We can find that the proposed scheme achieves high stability in both numbers of layers and throughput for both multicast sessions. This demonstrates that the congestion is controlled very well with the proposed scheme.

Fig 3.13 Individual TCP throughput and average per-flow TCP throughput in static (VBR).
In static variable bit rate (VBR) scenario there is no node movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The above figures in Fig 3.13 shows the throughput of each individual TCP flow and the average per-flow throughput of TCP flows, which specified the throughput at different levels. As shown in these figures, each unicast flow, gets a constant throughput till end of simulation.

We can find that the proposed scheme achieves high stability and fairness in throughput for all TCP sessions. This demonstrates that the congestion is controlled very well with the proposed scheme.
Random Node Movement Scenario

Fig 3.14 Individual multicast throughput and number of layers in random (VBR).

In random variable bit rate (VBR) scenario all the nodes are moving it means that both the source and destination have a movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The above figures in Fig. 3.14 shows the throughput of two multicast flow and number of layers for each multicast flow which is used to reduce the congestion, hence in first multicast flow (MCT1) when we start sending packets at 100 seconds we see the throughput till 800
second after that there is no throughput till the end of simulation because the nodes are moving it may not be in the range with any other nodes at that time and we have specifically defined the node range to be within 100 meters to communicate. There for even the number of layers is not used during this period of time because the layers are created depending up on the throughputs in the first multicast flow (MCT1), with respect to second multicast flow (MCT2) since there are two flows considered with different sources and one source may be moving away but other source still in the range, even in (MCT2) also the node is out of range for small amount of time after that a throughput which is available in second multicast flow even the number of layers also available.

As shown in Figure, the two multicast flows do not have a stable number of layers anymore as random movement is introduced to nodes in the network. This is because when nodes move, link quality and actual capacities change in the network.
Fig 3.15 Individual TCP throughput and average per-flow TCP throughput in random (VBR).

In random variable bit rate (VBR) scenario all the nodes are moving it means that both the source and destination have a movement, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless network) and stop at the 1,500th second.

The above figures in Fig. 3.15 shows the throughput of each individual TCP flow and the average per-flow throughput of TCP flows, which specified the throughput at different levels, in all three TCP flows we have near constant throughput, even in some time the throughput
changed this because of node movement because when nodes move, link quality and actual capacities change in the network, the TCP_Average graph shows the average throughput which is calculated for TCP1, TCP2 and TCP3.
3.3.4 Comparison between proposed MCAODV and existing protocol

In this section, we show the comparison between the proposed protocol with the Receiver-driven Layered Multicast (RLM)[16]. RLM is one of the existing protocols which is the most popular multilayer protocol for multicast congestion control. We only show the comparison results for the least challenging case, in which the multicast traffic is constant bit rate traffic and all nodes are static.

Fig 3.16 Comparison of multicast flow throughput and number of layers between MCAODV and RLM protocol (MCT Session-1)
Fig 3.17. Comparison of multicast flow throughput and number of layers between MCAODV and RLM protocol (MCT Session-2)

The above figures in Fig. 3.16 and 3.17 shows the comparison between the proposed protocol (MCAODV) and existing protocol (RLM), in two different multicast flows throughput and number of layers. We can find that the proposed scheme achieves higher stability in both number of layers and throughput for both multicast sessions. This demonstrates that the congestion is better controlled with the proposed scheme.
Fig 3.18 Comparison first and second TCP flow throughput between MCAODV and RLM protocol

In Figure 3.18 and 3.19, a comparison between our proposed protocol (MCAODV) and existing protocol (RLM), in individual TCP flow and the average per-flow throughput of TCP flows is given. In both figures, we can find that the proposed scheme achieves higher stability and fairness in throughput for all TCP sessions. This demonstrates that the congestion is controlled very well with the proposed scheme.
Fig 3.19 Comparison third TCP flow throughput and average per-flow TCP throughput between MCAODV and RLM protocol.

We have presented MCAODV protocol to support multicasting in wireless networks that also ensures unicast flow shares of bandwidth on a link. Existing routing protocols for multicasting in wireless networks such as MAODV and ODMRP, like unicast routing protocols, only set up routing information in nodes. They cannot control other characteristic of flow control such as congestion. There have been extensive efforts for creating multicast transport and congestion control protocols but they do not ensure fairness with TCP. Wireless networks pose more serious
challenges for those schemes because wireless networks have limited bandwidth, significant channel access delays, and high link error rates. Instead of relying on end-to-end schemes for supporting multicasting in wireless networks, this work proposes a localized scheme that protects unicast flows from being throttled by multicast flows in wireless networks. The work integrates layered multicast and routing-based congestion control for providing better throughput. In Static Network Scenario high stability is observed in both number of layers for multicast sessions and fairness in throughput for all sessions indicating the success congestion control mechanism.