Appendix A

Code 1: Identify_Channel_Controller()

Route_Add(Node_ip) //add in Route with all Ip address
//which has been registered in DC
Route_IP = Identify_Idle_Node(Route_Address [])
//Identify Idle node, add to Route
Route_SD = Identify_Source_or_destination_Node(Route)
Route_H = Identify_Route(Route)
//collect A_Ip, B_Ip, A_to_B_Available_bandwidth,
//and delay from QoS_Reservation_Manager.Get_Route()
If (Available_Bandwidth >= rBwd && uQoS > eQoS)
{
    //rBwd--decide by Service Discovery Manager
    //if the available bandwidth >= required bandwidth;
    //uQoS-User QoS; eQoS-expected QoS
    RouteQueue [v1] [v2] = delay // link delay (RTT)
}
else
{
    RouteQueue [v1] [v2] = α // α-session
}
CC [ i] = COAAS_Channel_Create (Route[ ], COAAS_QoS, Bandwidth)
//add Route bandwidth, QoS and Bandwidth matrix
//Channel create an algorithm to define channel
if (route[i] = = null)
{
    route [j] = COAAS_Channel_Assign (CC [Ii], Bandwidth_Avai (Route[ ])
    if(route[k] = = null)
    {
        //path2 not found
        route[k] = COAAS_Channel_Assign (RouteQueue,(Route_I + Route_SD + Route_H))
        //Source destination nodes Route and neighboring node
        if(route[k]= = null)
        {
            REP = ‘‘sour+stream_To_DC_Ip+s_portno’’
            Send (REP)
            //send message to the source node to transmit stream to DC_Ip
            REP = ‘‘Dest+stream_From_DC_Ip+d_portno’’
            Send (REP)
            //send to destination node to receive streams from DC_Ip
        }
    }
    else
    {
        route = path3
    }
}
else
{
}
} //path3 found
route = path2  //path2 found
}
else
{
    route = path1  //path1 found
}
if (route!= null)
{
    //route found
    if (Mid = = 0)
    {
        //not multicast communication
        REP = ‘sour + stream_To_HF_Ip + s_portno’  //REP - Route Reply
        Send (REP)
        //send Route reply to source node to transmit stream to HF_Ip
        REP = ‘Dest + stream_From_HF_Ip + d_portno’
        Send (REP)
        //send to destination node to receive streams from HF_Ip
        REP = ‘HndF + From_S_Ip + s_portno + To_D_Ip + d_portno’
        Send (REP)  //send to Hand-off node to receive streams from
        //Source_Ip and send stream to destination_Ip.
        Node_Routing_Table.add (Source_Ip, HF_Ip, dest_Ip)
        //update in the routing table
    }
    else if(Mid = = 1)
    {
        //multicast communication
        REP = ‘sour + Multicast_Ip + s_portno’
        Send (REP)
        //send Route reply to source node to send stream to Multicast_Ip
        for (I = 1 to No._of_dest_node)
        {
            //one or more destination node
            REP = ‘Dest + stream_From_HF_Ip + d_portno’
            Send (REP)
            //send to destination node to receive streams from new node
            REP = ‘Multicast_Ip + s_portno + To_D_Ip + d_portno’
            Send (REP)
            //send to Hand-off node to receive streams from
            //Multicast_Ip and send stream to destination_Ip
            Node_Routing_Queue.add (Source_Ip, dest_Ip)
            // update in routing queue
        }
    }
}
Code 2: COAAS_Policy()

Broadcast COAAS_COAAS_RREQ:
COAAS_RREQ (SDC_Ip, Bsour_Ip, Dest_Ip, rBwd, plp, Mid)
//SDC_Ip-Source DC_Ip which broadcast COAAS_RREQ,
// Bsour_Ip-source node send stream to destination in different
//domains (some time DC may be as Bsourse), Dest_Ip-destination
//node which needs to find in different domains, rBwd-required minimum
//bandwidth for service, plp-packet loss percentage, Mid-multicast id
Receive COAAS_RREQ:
    If (DCi_DCj_Available_bandwidth >= rBwd && plp <= α)
    {
        // DCi which broadcast COAAS_RREQ, DCj - DC which received
        // COAAS_RREQ, if DCi to DCj available bandwidth >= required
        // bandwidth and packet loss percentage <= α.
        flag = false
        cDt = Check_in_DC_database(dest_Ip)
        //check destination in the database
        if ( cDt = = 1)
        {
            flag = true
        }
        //destination node is in the domain
        if (dest_Ip.equals (Localhost_Ip))
        {
            flag = true
        }
        //DC as destination node
        if (flag = = true)
        {
            COAAS_RouteTable.add (SDC_Ip, seqno, hop_count, next_DC_Ip, ‘’A’’)
            //add in COAAS routing table ‘’A’’ - Alive route
            COAAS_RREP = SDC_Ip + DtDC_Ip + Bsour_Ip + Dest_Ip + seqno + hop_count
            //DtDC_Ip - destination DC Ip which means the destination node
            //being in destination DC domain, sequence no, and hop count
            Send (COAAS_RREP) //send route reply message
        }
        else
        {
            COAAS_RouteTable.add (NDC_Ip, seqno, hop_count, NDC_Ip, ‘’A’’)
            //establish reverse path NDC_Ip-Neighbor DC
            brCOAAS_RREQ (SDC_Ip, Bsour_Ip, Dest_Ip, rBwd, plp, Mid)
        }
    //broadcast COAAS_RREQ
    else
    {
        COAAS_RouteTable.add (NDC_Ip, seqno, hop_count, NDC_Ip, ‘’A’’)
        //establish reverse path NDC_Ip-Neighbor DC
        brCOAAS_RREQ (SDC_Ip, Bsour_Ip, Dest_Ip, rBwd, plp, Mid)
    }
    //broadcast COAAS_RREQ
Receive COAAS_RREP:
    if (SDC_Ip.equals (Local host_Ip))
{ 
//if Source DC is local host if there is two or more 
//COAAS_RREP came from different DC, back up that route 
//without rejecting them, it will use for optimal route manager
COAAS_RouteTable.add (dest_Ip, seqno, hop_count, NDC_Ip, ‘‘A’’)
COAAS_RouteTable.add (dest_Ip, seqno, hop_count, NDC_Ip, ‘‘B’’)
//Two or more COAAS_RREP has back-up. ‘‘B’’ - Back-up route
Data = SDC_Ip + DtDC_Ip + Bsour_Ip + Dest_Ip + Portno + seqno
//send stream port no as a data to the destination node
Send (Data) //send data to destination DC
}
else
{
COAAS_RouteTable.add (NDC_Ip, seqno, hop_count, NDC_Ip, ‘‘A’’)
//establish forward path NDC_Ip-Neighbor DC forward
//(COAAS_RREP) Forward COAAS_RREP to neighbor DC.
Receive Data:
if (DtDC_Ip.equals(Local host_Ip)
{ 
//if destination DC is a local host
DatAck = SDC_Ip + Bsour_Ip + Dest_Ip + seq no
Send (DatAck) //Send data Acknowledgment to Source DC
if (!(dest_Ip.equals(Local host_Ip)))
{
//not DC as destination node
COAAS_RREP = ‘‘Dest+stream_From_DC_Ip + d_port no’’
Send (REP)
//send to destination node to receive streams from DC_Ip
}
}
}

Code 3 : COAAS_Resource_Discovery_Manager ( )
Receive (REQ)
// receive REQ message sent by the nodes for transmitting stream.
// Collect type of service, source and destination address, multicast id.
rBwd = Service_Discovery_Manager(type)
cSo = Check_in_DC_database(Source_Ip) // check source in the database
cDt = Check_in_DC_database(dest_Ip) //check destination in the database
if (cSo = =1 && cDt= =1
{}
//source and destination exist in the database
Find_Route_Algorithm (Source_Ip, dest_Ip , rBwd, plp, delay,
Qos_Reservation_Manager.get_Route(), Mid) //Mid-Multicast Id
} else if (cSo= =1 && cDt= =0)
{
// destination not exist in the database
COAAS.Activate (Source_Ip, dest_Ip , rBwd, plp, delay, Mid)
}
// activate COAAS to find and establish a route with
// destination node in different DC

Code 4: COAAS Execution ( )
Step a:
   Is COAAS_Node_Status= ‘ACTIVE’ and Route_Status=‘TRUE’
   Add_Route (COAAS_Route [ ])
   else
   Refresh_Route ( )
Refresh_Request ( )  // request issued by the controller
Step b:  // determine QoS congestion value
   Is COAAS_Route > COAAS_QoS_Value (Service [ ])
   {
      Update_Route ( )
   } else
   Step a, c;
Step c:
   Is Traffic_Type ( ) > COAAS_Traffic_Value and Service_Type ( ) > COAAS_Value
   {
      // check for priority of the route
      Assign_Route:= Traffic_Priority_Handler ( )
      else
      Assign_Route:= Normal
      Assign_Route_Intensity:= COAAS_Weight
   }
   Update_Channel_Coordinator (COAAS_Route [ ])
Step d:
   Channel_Create (CC [ ])
   {
      CC (Node_Bandwidth, COAAS_Route [ ], Traffic_Type,
      COAAS_Weight, Service_Type [ ])
      Update_Channel (CC)
   }
Step e:
   Call COAAS_Policy_Manager (CC [ ])

123