# Context Aware Adaptive Service based Dynamic Channel Allocation Approach for Providing an Optimal QoS over MANET

A. Ayyasamy[1], K. Venkatachalapathy[2]
[1, 2] Department of Computer Science and Engineering, Annamalai University, Tamilnadu, India.
[1] samy7771@yahoo.co.in
[2] omsumeetha@rediffmail.com

**Abstract—Large variations in network Quality of Service (QoS) in terms of bandwidth, latency and jitter may occur during media transfer over mobile ad-hoc networks. Applications need to adapt their functionality according to dynamic change of their QoS update. This paper proposes an enhanced service based platform to provide adaptive network management services to higher level application layer components. The Context Aware Adaptive Service (COAAS) is a middleware architecture for service adaptation based on ad hoc network and service awareness. COAAS is structured in such a way that it can provide QoS awareness to streaming applications as well manage dynamic ad hoc network resources using an adaptive channel allocation approach. The overall architecture of COAAS framework includes core components to connection establishment, connection monitor, connection controller and policy manager. Adaptive channel allocation defined as object based component helps in dynamic binding during run time implemented using JXTA and J2ME using CDC [15] toolkit to demonstrate the performance of a mobile setup as a conference application.**

**Keyword-** Adaptive channel allocation, COAAS, QoS, MANET

## I. INTRODUCTION

Optimal Quality of Service (QoS) depends on the underlying network communication infrastructure to provide access to multiple services and managing resources [24]. Ideally, such QoS critical applications do not have any concern anything about the networks used since they focus on the service functionalities. Large variations in network QoS such as bandwidth, latency, jitter and reliability may occur during media transfer over ad hoc networks [16], which degrade the performance of service. In this paper, an optimal QoS scheme Context Aware Adaptive Service (COAAS) over Mobile Ad-hoc Networks (MANET) is proposed, which is a middleware architecture, which focus on identification of the optimal service quality metrics [13] and adaptive bind up during dynamic runtime environment [10], [17]. The primary two objectives of providing methods to achieve optimal QoS with dynamic stateless routing among MANET as the major phenomenon and mechanism to identify and support in dynamic channel based on QoS requirement and service applications.

Mobile ad-hoc networks [2], [5] are highly dynamic in terms of available mobility, session management related to network resources, connectivity, location management and heterogeneous devices such as Bluetooth, IrDA, Wi-Fi or WLAN, etc. Traditionally, middleware [6], [10] is required to support heterogeneity and to enable the application programmer to focus on application issues. The research work proposes to develop a middleware service that additionally provides quality services for information sharing in MANET, since the possibility to share information is mission critical for many mobile ad-hoc network applications. The aim is to identify solutions for this realistic setting and to quantify the QoS [3] which should support multiple service based approaches to users.

The following objectives are addressed:

a). To design a dynamic service based channel allocation approach for providing variable service based support over MANET.

b). To support MANET distributed node location and routing management using QoS supportive middleware architectural approach.

c). To deal with the dynamic state of the mobile node during routing, corresponding with available resources based on service in use.

d). To provide predictable mission-critical end-to-end QoS services and a mechanism which plays a major role in any distributed real time system which works under unpredictable situations.

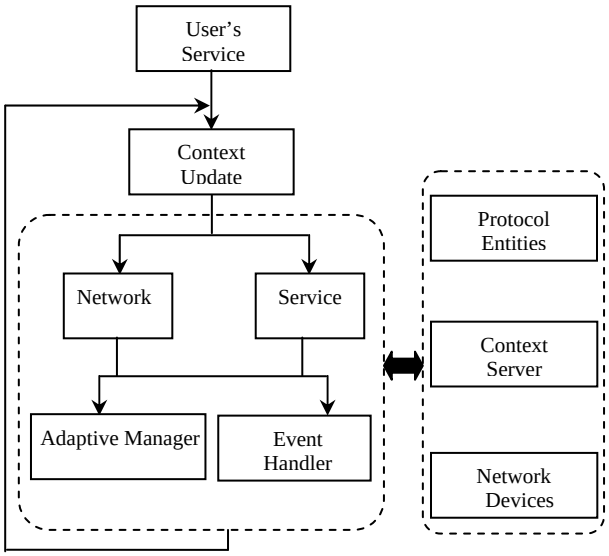


Fig. 1 COAAS architecture and middleware functionality

This paper provides two contributions to the study of adaptive middleware to control distributed network, service and user's real-time requirement. It describes how priority [19] and resource reservation[4] based network QoS management mechanisms can be coupled with MANET standards using off-the-shelf Distributed Object Computing (DOC) middleware [12], [16] to better support dynamic DRE applications with stringent end-to-end service based computational real-time requirements as shown in Fig. 1.

*A. Motivation*

Mobile devices are heterogeneous in communication setup [11] with variable support for network connectivity [2]. These devices vary in terms of processing, input / output capacities, energy consumption, and session establishment. Node mobility [9] leads to the continuous change in location, environment, network provider and access networks. As the new computing paradigm for the next generation mobile computing [8] pervasive computing [1], [7] introduces more variations to network performance where the communication technologies could be highly diverse and overlapping over a large space. To specify a user terminal may be equipped with multiple connectivity technologies ranging from wireless mobile network and short-range ad hoc connection up to local and wide range connections. Most of the existing QoS models [21], [26] focus on network supportable parameters such as bandwidth, latency, and jitter targeting to provide a transparent quality support on transport systems to upper applications. However, in such approaches, the mechanisms to support adaptive user demandable resource reservation [22] are neither sufficient nor feasible. With an increasing need for network applications to be aware of variation in network performance [20], [24] and quality [4], the applications should be highly "context aware" [18], [5] in order to adapt to change in multiple network environments and services.

In our model assumes that multiple nodes communicate with each other, based on a set of resource awareness policies [16] to establish an optimal QoS route among multiple nodes engaged in session as shown in Fig.1. The proposed set of QoS services considers service and user based policies for sharing resources, hence tailoring the network domain flexibly. Resource awareness services [22], [23] are layered upon a set of communication services in a middleware architecture enabling communications between nodes belonging to distinct ad-hoc networks. We proposed COAAS middleware architecture for adaptive applications based on context awareness. The overall architecture of COAAS emphasizes service based QoS and network management context awareness over user and resource utilization.

The rest of the paper is organized as follows: Section II presents the overall architecture of COAAS. Section III discusses the context considered in COAAS and the methods of how to realize context awareness. Section IV discusses the realization of adaptive network supports and its utilization on service adaptation. Section V concludes the paper with a discussion of future work.

## II. COAAS ARCHITECTURE

Context aware adaptive QoS middleware support over MANET networks focuses on providing end to end QoS over service and resource availability. COAAS works on a set of policy manager which entitles definition of multiple dynamic channels being assigned as per service in use.

### A. *Architecture and Execution Environment*

The COAAS architecture defines dynamic channel allocation 'Ci' for service, network and the expected user's QoS by using well defined policy sets. Services in use are defined at run time through objects space [27], which binds to event functionality [24] for exhibiting their adaptive behavior along with network Operating System (OS) and related kernel components [28]. The networking components and underlying infrastructure support heterogeneous OS, network and sub-network domain setup as shown in Fig. 2.

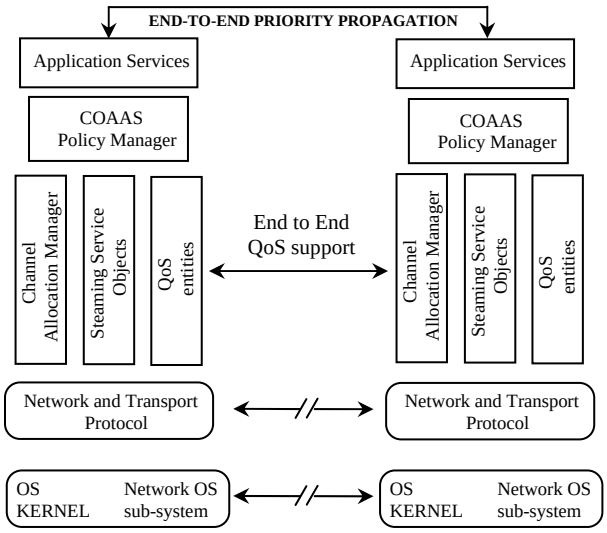


Fig. 2 COAAS: End to End QoS (runtime environment)

COAAS middleware infrastructure defines a five layered stack architecture which functions on object monitoring, control and query of device status [20] with extended services towards session establishment. Network devices [23] include various network components such as network adapter card, modem, access point, routers and gateway. Hence, gathering the related system physical configuration is primary to control the network. These physical entities are highly variable for any service, but helps in defining the variable QoS format for temporary service in use and user defined QoS. Protocol entities include interfaces for the management of network device drivers, protocol stack and routing table. Network contextual information can be locally positioned in an end host and maintained in a distributed setup. COAAS middleware is identified as a software platform above the operating system and other resource infrastructure to provide adaptive network connectivity management to upper system modules and services. COAAS's user defined policy adapts the service and node such that the network's QoS utilization is optimal and negotiable as per the available resources. The detailed processing of the adaptation demands is left to connect controller without any concrete concern or update from the applications.

Fig.3 shows the general COAAS architecture and execution environment. The COAAS policy manager and policy administrator nominate an adaptation function which relates policy set to service in use. The adaptation of supported service can be realized through various sessions in use along with measurable QoS components at different stages, including service adaptation triggering, network resources selection and binding the objects at run time along with the policy. Adaptation mechanisms are first triggered by some specific context according to the predefined matching criteria. A decisive policy is arrived which keeps the resource components being updated for any small changes and maintains the adaptation approach. The service adaptation is achieved by automatically or manually executing a command and/or changing the external behaviors (and possible internal states) of an entity that provides the service. COAAS provides the adaptive framework using a set of network APIs (SMTP, FTP etc.,) to the upper network-aware adaptive applications are seen in Fig. 2. The COAAS's policy administrator function deputes services from primary service requirement as abstract components, but as well performs OS kernel functionalities [17] and resource management as middleware layer tasks. The adaptation of the end application starts from the transport layer to the service based application layer as the middleware functionality. All the network adaptation mechanisms are abstracted and represented as objects onto the COAAS middleware level, since the monitoring and control of available network resources are most convenient to be implemented at this level. Semantic oriented adaptation mechanisms are finally decided at the

application level for the play-out sequence and handle jitter. The fact that service updates its content and media, which it consumes and processes the best from physical system objects to service object components.

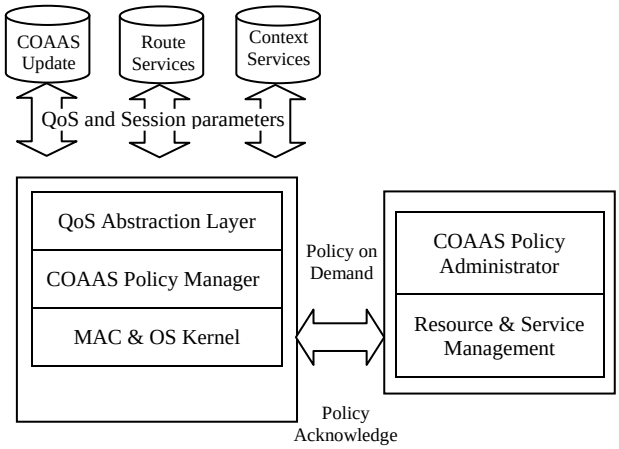


Fig. 3 COAAS architecture and execution environment

*B. Dynamic Channel Allocation*

Dynamic channel and connection controller forms the management core of COAAS middleware. The service management realizes a service as a in use by the object creation and maintenance of the connection channel [10]. A channel is defined as the logical link or connection that exists between any two communications, peer entities or service application components of COAAS architectural stack, which are defined between various network devices e.g. terminal or a server. Each channel uses an end to end service specific connection to transfer data between multiple devices. The session established as a channel along with user specific QoS variables can be dynamically changed, while leaving the channel unchanged and hence making applications imperceptible. Fig. 4 shows the collaborative operation among nodes where COAAS maintains the optimal selected route based on available QoS and resources in use. Managing the connection channels is the core function of COAAS, which is realized by connection controller. Multiple channels connecting to different nodes should be under the control of the COAAS in the host. The differences of the two operation modes can be perceived by COAAS and not by services.

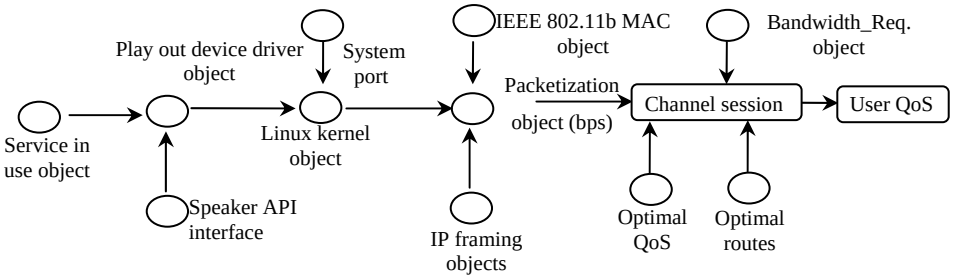


Fig. 4 Channel connection session defined as objects

Channel Definition $Ch_i$
Channel Assignment $\{\alpha\ \beta\ \rho\}$ …. (1)
$\tau$ = Channel Revoke $\{\tau\ n\}$ …. (2)
Channel Update $\{\tau\ \alpha\ n\}$ …. (3)

$\alpha$ Service in use
$\beta$ Bandwidth required
n Bandwidth supported
$\tau$ Route in use
$\rho$ Expected delay

Channel, Ch defined in equation 1 where 'I' being the channel number which is assigned equation 2 with dynamic variables such as bandwidth in use, type of service and delay expected (as QoS parameter) for a service to be supported. The channel is revoked equation 3 or re-created with an updated route and bandwidth supported based on session update.

*C. Channel Operations*

1) *Channel maintenance*:

Channel connection information maintained by connection controller assists the maintenance of service based on network session established. The channel controller maintains the lists of the references of all the interfaces, channels and policies, along with the mappings between them. The resource lists and mappings are continuously updated in case of any special event (e.g. a channel has switched the connection under using or a new channel is opened with a new policy).

*2) Channel Update:*

The core adaptation mechanism of COAAS architecture is realized using a connection controller through adaptively maintaining connection channels. There are two activities of the connection controller concerning channel maintenance, i.e. channel opening and switching. To open a new channel application may provide four parameters: target host name, traffic class, channel direction, and policy set. Connection controller then queries the nodes in neighborhood using target host name.

3) *Channel Revoke / Switching:*

Connection controller periodically re-evaluates the mappings between an interface and each channel according to the policy used for each channel. Moreover, the revoke is carried out when events such as interface up or down, channel opened or closed take place. If, according to the policy, a better interface is found, then the connection controller initializes a channel switching session. The session needs the cooperation between controller peers through the signalling channel, as shown in Fig. 5. The phenomenon of revoking happens when any incoming and outgoing connections are being decided by host node. The decision is carried out based on contextual information of service in use and QoS value defined.

*D. Context Awareness and Connection Monitor*

The context awareness works primarily based on network management and adaptive service on a user using components such as connection monitor. Connection monitor gathers traffic from channel consistently using a polling approach [18] or querying approach and verifies the organization of network configuration and related contextual information. This information is primarily used by services and connection controllers for implementing normal network supported functions and adaptation. Both local and end-to-end network information can be monitored. The contextual network information includes:

a). The interface information is interconnected virtually across all the network interfaces as host, using corresponding host name, domain name, DNS servers and node type.

b). Number of network interfaces, as the fact that a multiple mobile device may be equipped with multiple network interfaces, e.g. Bluetooth, IrDA, modem, Ethernet, WLAN and Wi-Fi.

c). Information of each network interface, which includes name, type, physical and IP addresses, gateway, DHCP server, speed, configuration parameters (e.g. Dial-up number, user account, password, etc.), the traffic workload at local interface and the access point being used, error rate, signal strength, SNR, power consumption, and operation status (e.g. Available, operable, connecting, connected, sleeping, idle, transmitting, receiving, unconnected, unreachable, disabled, etc.).

d). Packet statistical information such as received, sent, and dropped packets of protocols of IP, ICMP, TCP, and UDP.

*E. Adaptation and Policy Manager*

The adaptation mechanism for the network management in COAAS is primarily realized through the policy manager and channel session controller as shown in Fig. 5. In COAAS model refers is identified channel controller in Algorithm 2. COAAS employs a policy mechanism to ease the adaptive management of network resources. The services suggest their adaptive requirements with policies by creating new channels, such that the policies are configured at run time by COAAS. Policy denotes the criteria for the selection of the optimal route setup such that the connection controller maintains each channel according to well defined policy.

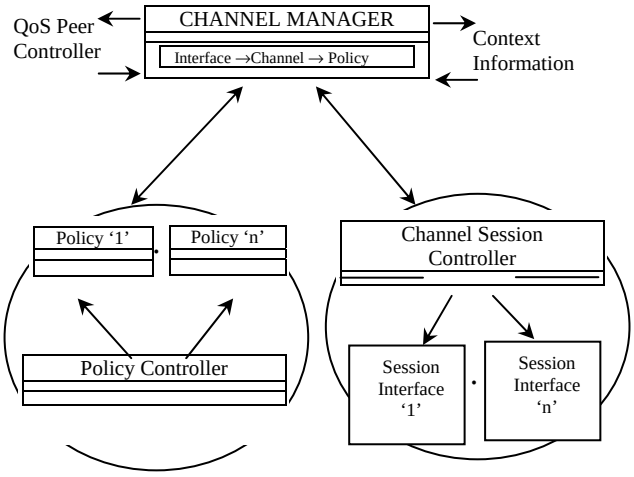


Fig. 5 COAAS channel manage

A policy manager updates and maintains the policy as either static or adaptive, where static policy explicitly declares the network interface to be used, while dynamic or adaptive policy defines at runtime the access selection rule for the variable specific type of traffic flow. An adaptive policy can be represented by traffic class, network logic and QoS weighted factors. Traffic class can be any value defined in TOS [21], TC [14], DS [25] or COAAS application specific value.

*F. COAAS Policy Manager*

Policy manager is used by applications to supervise policies, which include policy creation and close. Policies are then accessed by connection controller during the channel operations in Algorithm 3. Some of the application policies may conflict with user preference policies stored previously in context repository. However, user preferences always have the highest priority. COAAS MAC queue manager analyzes in Algorithm 1 and determines any contextual change in the execution properties of the network, which adapts the channel manager to create due to change in requirements of the network.

Inclusion of dynamic context aware parameters such as service average throughput over a variable period of time, service delay and node to node connectivity play a major role. The instance of route scheduler state, packet incoming / outgoing state and service admission state derive the system to be adapted for any change in service. This scheme was implemented over MAC and network layer of MANET routing protocol. The presence of multiple mobile gateways and the maintenance of prioritized service based routes have increased connectivity and session availability. The algorithm3 works on admission of service or packet into a scheduled active queue. The instance of a packet reaching the weight of drop after an interval of time signifies the chance of system degradation.

**Algorithm 1: COAAS_Resource_Discovery_Manager ( )**

```
Receive (REQ)          // receive REQ message sent by nodes for transmit 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 Database
  cDt = Check_in_DC_database(dest_Ip)       //check destination in Database
if (cSo= =1 && cDt= =1
{ //source and destination is exist in database
Find_Route_Algorithim (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 database
        COAAS.Activate (Source_Ip, dest_Ip , rBwd, plp, delay, Mid)
} // activate COAAS to find and establish route with // destination node in different DC
```

**Algorithm 2: 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 ()
  // generate matrix.  Each node consider as vertex. The delay consider as edge weightage.
            If (Available_Bandwidth >=  rBwd  &&  uQoS >  eQoS)
 {          // rBwd –decide by Service Discovery Manager
             // if 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 algorithm to define channel
            if (route [ i] = = null)
 {                                                        //path1 not found
route [j] = COAAS_Channel_Assign (CC [Ii], Bandwidth_Avai [ Route_[  ] )   //add Graph matrix and
combined (Idle nodes Route and Source
            //destination    notes route) in algorithm to define channel
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){                                   //path3 not found
REP = "sour+stream_To_DC_Ip+s_portno"
Send (REP)          //send message to source node to transmit stream to DC_Ip
REP = "Dest+stream_From_DC_Ip+d_portno"
Send (REP)          //send to destination node to receive stream from DC_Ip
 } else {route = path3}                                  //path3 found
} else {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 stream from HF_Ip
REP = "HndF + From_S_Ip + s_portno + To_D_Ip + d_portno"
Send (REP)          //send to Hand-off node to receive stream from Source_Ip      //and send stream to
destination_Ip. Node_Routing_Table.add (Source_Ip, HF_Ip, dest_Ip)
//add in routing table
} else if(Mid = = 1){                                   //multicast communication
 REP = "sour + Multicast_Ip + s_portno" //Eg. Multicast_Ip = 230.0.0.3
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 stream from new node
```

```
    REP = " Multicast_Ip + s_portno + To_D_Ip + d_portno"
Send (REP)        //send to Hand-off node to receive stream from Multicast_Ip      //and send stream to destination_Ip
Node_Routing_Queue.add (Source_Ip, dest_Ip)  // add in routing queue
} } }
```

**Algorithm 3: 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 domain (some time DC may be as Bsource),
//Dest_Ip- destination node which need to find in different domain, rBwd –required
//minimum bandwidth for service, plp-packet loss percentage, Mid – multicast id
```

*Receive COAAS_RREQ:*

```
   If (DCi_DCj_Available_bandwidth >= rBwd && plp <= 25)
{          // DCi which broadcast COAAS_RREQ, DCj – DC which received COAAS_RREQ,  //if DCi to DCj available bandwidth >= required bandwidth and packet loss percentage <= 25.
  flag = false
  cDt = Check_in_DC_database(dest_Ip)                  //check destination in Database
  if ( cDt = = 1) { flag = true}                       //destination node is in 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
```

*Receive COAAS_RREP:*

```
if (SDC_Ip.equals(Localhost_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 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(Localhost_Ip) {          //if destination DC is local host
                DatAck = SDC_Ip +  Bsour_Ip + Dest_Ip + seqno
```

```
            Send (DatAck)                    //Send data Acknowledgement to Source DC
            if (!(dest_Ip.equals(Localhost_Ip))) {          //not DC as destination node
            COAAS_RREP = "Dest+stream_From_DC_Ip+d_portno"
            Send (REP)          //send to destination node to receive stream from DC_Ip
  }
 }
}
```

**Algorithm 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 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 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 [ ])
```

*G. COAAS Execution*

Channel management and controlling are the core components of COAAS for the final realization of assigning a resource or utilization of network management mechanisms as shown in Fig. 6. Algorithm 4 explains the COAAS execution using a set of policies and service components. The contextual information of mobile nodes, route in use, application traffic at network controller and required QoS defines the channel. At the same time it is also the entity for the interaction and cooperation with other related components using XML contents [6] and peer controllers. The COAAS functionality, architecture adopts two categories, as context information and channel management.

Essential parameters of the experiment include the following:

1) Network size - Number of nodes

2) Network connectivity - Average degree of a node (average number of neighbors of a node)

3) Topological rate of change - The speed with which a network's topology is changing

4) Link capacity - Effective link speed (bits/second), after accounting for losses due to multiple access, coding, framing, etc.

5) Fraction of unidirectional links -Effectiveness of protocol performance as a function of the presence of unidirectional links.

6) Traffic patterns - Protocol effectiveness in adapting to non-uniform or bursty traffic patterns.

7) Mobility - An instance of mobility such that the circumstances are temporal and spatial topological correlation relevant to the performance of a routing protocol.

8) Fraction and frequency of sleeping nodes - COAAS protocol performance in the presence of sleeping and awakening nodes.

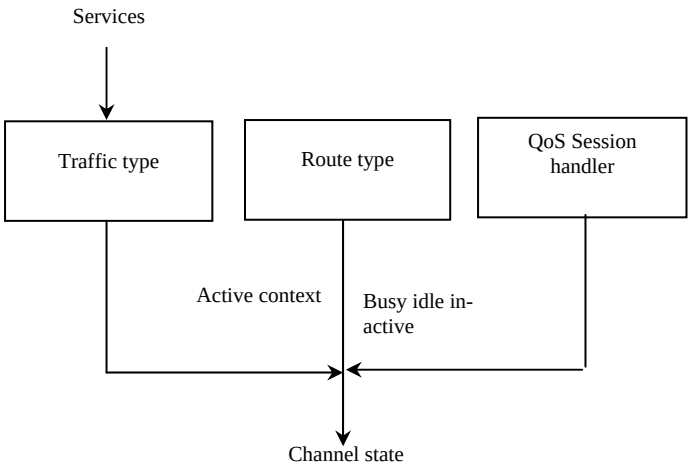


Fig. 6 Functional diagram of COAAS

## III.EXPERIMENTAL APPROACH

The outdoor routing experiment was carried out on a rectangular athletic field (200 (north-south) x 300 (east-west) meter as shown in Fig. 7. The traffic generator on each mobile node generated packet streams with a mean packet size of 1200 bytes (including UDP, IP and RTP headers), a mean of approximately 5.5 packets per stream, and a mean delay between streams of 15 seconds. These parameters, generate an approximation of 423 bytes of data traffic (including UDP, IP and RTP headers) per node per second, with fair traffic volume, but corresponding to the traffic volume observed during trial runs as one of a prototype media streaming applications. All four algorithms are implemented in JXTA using a core set of classes. These classes include the event loop, as well as unicast and multicast, routing, and logging support.

A. *Hardware Platform*

Experiments were conducted by using varying set of 25 Lenovo nodes, over differing environment such as IEEE 802.11 standards of a, b/g, n MANET interface, 128MB of main memory, Intel Pentium III processor, Intel Dual-core, I3 and I5 chipsets. A coordinator node is used to control each experiment, and leaving another 24 nodes to create and run the COAAS QoS and routing algorithms. The nodes run on Linux, Windows OS with PCMCIA, as well as can transmit data at variable bit rates. It can also auto-adjust the bit rate depending on the observed signal-to-noise ratio. Dynamic channels help to arrive at a consistent rate, channel for all the nodes in the network. The nodes are implemented in "ad hoc mode" setup in which the transmission rate was fixed at 2 MB/s to 54Mbps such that the channel can automatically choose the setup. Specifically, the setup used was Lucent (Orinoco) firmware version 4.32 and the proprietary ad hoc "demo" mode suggested by Lucent.

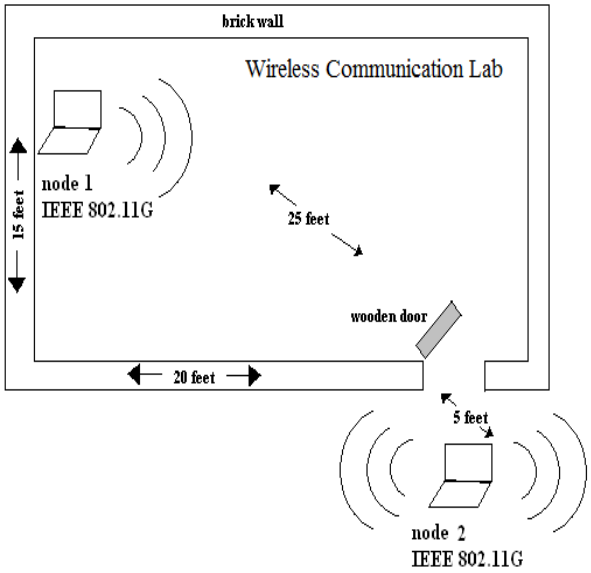


Fig. 7 Outdoor experimental test-bed

To ensure consistency with multiple series of ad hoc routing experiments "demo mode" is suggested, such that the outdoor experiment is a culminating event. The fixed rate of transmission makes it easier to analyze the routing results and the need to account for automatic changes in each card's transmission rate is accountable. The multi-rate capabilities and their general improvements over the demo mode are proposed to suggest variable bit rate traffic. As the demo mode provides sufficient functionality to serve as a reasonable data-link layer, the routing results determine as a representative. Each node is also attached to the Garmin General Positioning System (GPS) unit through the serial port to support an accuracy of thirty feet throughout the experiment. The experimental test bet result has been shown in table 1.

TABLE 1
Outdoor statistics gathered

| Models | Message Delivery Ratio | Data Packets Per Message | Control Packets Per Message | Total Packets/ Message | Average Hop Count (successful messages) | Message Latency (seconds) |
|---|---|---|---|---|---|---|
| **AODV** | 0.20 | 1.32 | 6.18 | 7.50 | 1.61 | 0.49 |
| **DSR** | 0.50 | 0.90 | 32.40 | 33.30 | 2.11 | 1.32 |
| **COAAS** | 0.77 | 22.79 | 22.80 | 45.59 | 2.47 | 0.37 |
| **DSDV** | 0.08 | 0.20 | 150.47 | 150.67 | 1.18 | 2.98 |

## IV. PERFORMANCE & EVALUATION

From the case study above, several conclusions can be drawn on the performance of the above mentioned routing algorithms. First, COAAS outperforms AODV significantly in terms of routing overhead in low mobility (small *p*) and small-scale network (small *N*) situation. However, its performance deteriorates rapidly when the situation gets stressful, i.e. *p* and *N* increase. This is due to the aggressive use of source routing cache. During a route discovery process, the source can learn several routes to its destination. This enables the source node to switch to cached routes in case of the currently used route break, which significantly reduces the possibility to restart a route discovery process. However, in stressful situations, it is more likely that all the cached routes are already invalid and thus there is unnecessary delay and extra network traffic.

TABLE 2
Experimental test-bet and results

| **Video Streaming** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Movie for Experiment** | **Average Bandwidth Required Mbps** | **Average Bandwidth Used Mbps** | **Multicast Groups / User** | **Delay ms** | **RTT ms** | **Hops** | **Loss %** | **Jitter ms** |
| **Movie –A 1800Mbps** | 1200 | | | | | | | |
| DSR | | 160 | 5 | 30.0 | 16.0 | 6 | 26 | 425 |
| COAAS | | 110 | 5 | 23.3 | 15.4 | 6 | 20 | 384 |
| DSDV | | 135 | 5 | 27.0 | 20.9 | 8 | 20 | 450 |
| AODV | | 210 | 5 | 35.4 | 23.6 | 12 | 31 | 502 |
| **Movie -B 2440Mbps** | 1220 | | | | | | | |
| AODV | | 220 | 7 | 39.0 | 15.7 | 7 | 42 | 521 |
| COAAS | | 197 | 7 | 31.5 | 15.2 | 5 | 34 | 504 |
| DSR | | 206 | 6 | 37.7 | 15.9 | 5 | 44 | 600 |
| DSDV | | 245 | 7 | 39.8 | 17.2 | 6 | 68 | 628 |
| **Movie-C 2200 Mbps** | 2000 | | | | | | | |
| AODV | | 232 | 6 | 31.0 | 14.2 | 6 | 31 | 500 |
| COAAS | | 204 | 6 | 28.8 | 13.6 | 5 | 30 | 478 |
| DSR | | 249 | 6 | 29.1 | 14.8 | 5 | 42 | 507 |
| DSDV | | 210 | 6 | 34.0 | 16.1 | 6 | 58 | 578 |
| **Movie-D 2300 Mbps** | 2990 | | | | | | | |
| AODV | | 201 | 7 | 36.6 | 15.1 | 5 | 38 | 510 |
| COAAS | | 165 | 6 | 31.0 | 14.0 | 5 | 21 | 464 |
| DSR | | 240 | 6 | 34.2 | 15.4 | 5 | 30 | 525 |
| DSDV | | 214 | 6 | 39.0 | 17.3 | 6 | 47 | 570 |

In the first set of experiments, the performance of COAAS with varying number of packets per source node is tested in Table 2. Since each mobile node creates a new packet with a fixed sampling interval, the total number of packets created by each node is determined by the duration of the experiment. The traffic test illustrates the basic performance properties of the protocol. It has to be noted that the default experimental settings are used in the traffic test, except the number of packets created per node variables in individual experiments. The experiment starts with generating 50 packets per node, which sums up to 250 packets in total in the network. Then the number of packets created (and the experiment duration) is increased by 40% to 100 packets per node, which equals to 500 packets in total. At last, the number of packets created is increased by 120% to 200 packets per node, which sums up to 1200 packets. The experimental results are shown in Fig. 8, Fig. 9 and Fig. 10.

The increase in the "route discovery" interval and "packet admission" rate degrades the behavior of the proactive route maintenance system, which occurs in a set of nodes closer to the scheduler gateway. Any spurious update of packet increase may cause congestion due to excessive traffic. In Fig. 10 shows the effect of registered nodes on the throughput, delay and gateway overhead.

a) If there is an increase in the percentage of registered nodes, the throughput decreases by 7% while the delay rises by 32ms.

b) If the number of registered node increases, a single mobile gateway can serve more nodes resulting in relatively lower throughput. The performance of DSR scheme was observed to be 12% better than the pure reactive scheme AODV.

In the node discovery scheme of COAAS scheme, as mobile gateways broadcast advertisements within periodic intervals, the overhead slightly increases. However, the overhead is higher by 3% in pure reactive scheme compared to the DSR schemes when the number of registered nodes increases. As the number of registered nodes increases, the mobile gateway overhead increases by 5.9%.

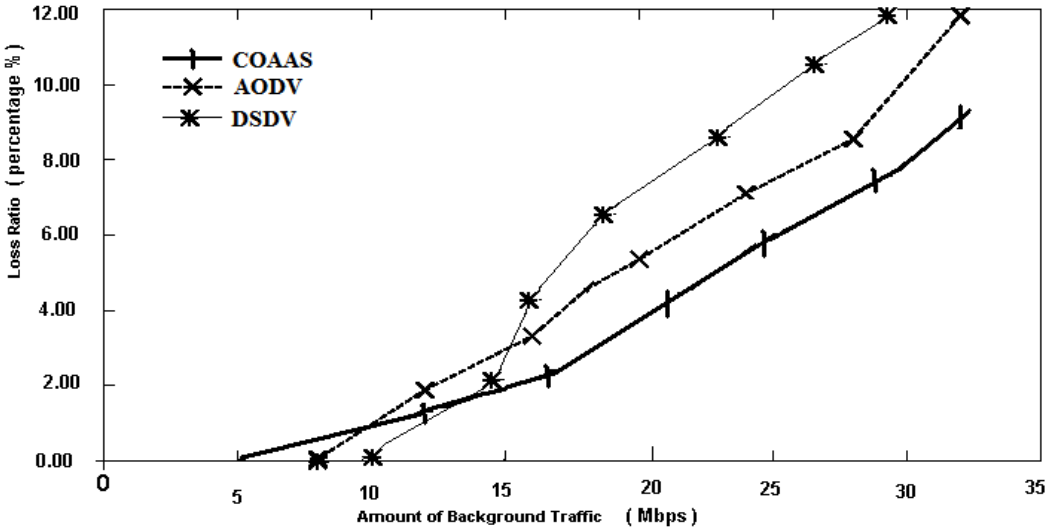


Fig. 8 Percentage of packet loss against traffic intensity

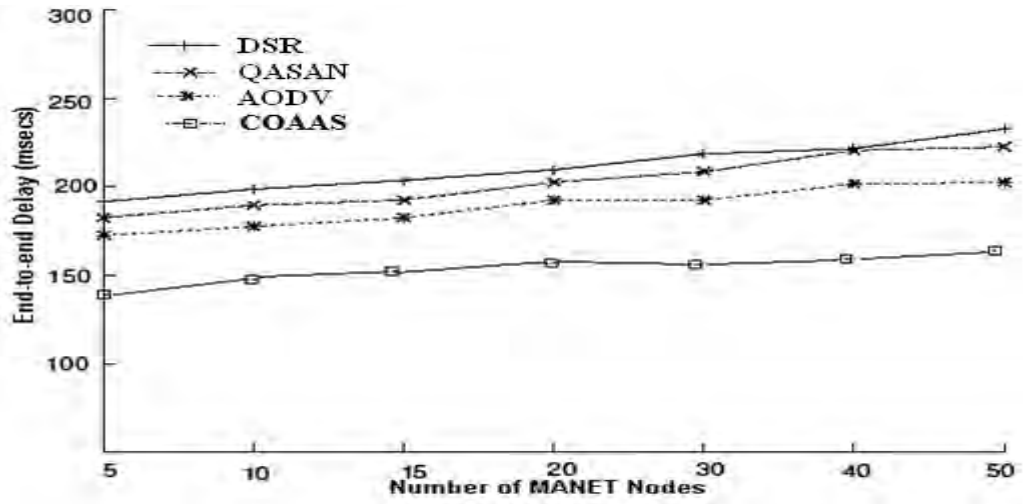


Fig. 9 End to end delay measured against number of MANET nodes

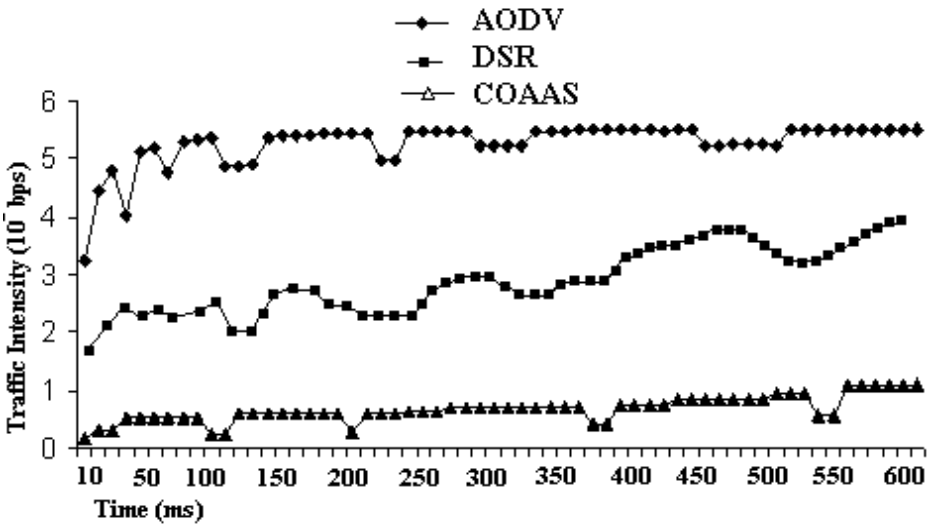


Fig. 10 Traffic load consumed against increase in time

## V. CONCLUSION

COAAS is necessary for the optimization of future mobile communications as a platform for intelligent services. The architecture proposed in this paper aims at providing an adaptive network management for QoS supportive network-aware applications. The mechanisms for providing service aware QoS using dynamic channel management for improving context awareness and adaptation are supported. COAAS is currently a work in progress, while it is implemented as a preliminary prototype based on Java on PDA's, pocket PC platform. More functionality will be implemented gradually during the COAAS's evolvement. Moreover, the future work of COAAS can be further extended by embedding the QoS into a broader platform of investigation of real context-aware applications. Future work can also focus on providing end to end security along with providing QoS.

## REFERENCES

[1] Abolhasan, M., Wysocki, T. and Dutkiewicz E., "A review of routing protocols for mobile ad hoc networks," Ad Hoc Networks, vol. 2, no. 1, pp. 1-22, 2004.
[2] Anuradha, S., Raghuram, G., Sreenivasa murthy, K. E. and Gurunath Reddy B. "New Routing Technique to improve Transmission Speed of Data Packets in Point to Point Networks," ICGST-CNIR Journal, vol. 8, no. 2, pp. 66-71, 2008.
[3] Arunkumar T. "QASAN: Delivering Quality of Services for media streaming services in Group Communication over Mobile Ad-Hoc Networks," in Proc. of the First International Conference on Industrial and Information Systems (ICIIS' 06), 2006, p. 435-443.
[4] Azzedine Boukerche, Begumhan Turgut, Ladislau Blni and Damla Turgut. "Routing protocols in ad hoc networks: A survey," Computer Networks: The International Journal of Computer and Telecommunication Networking, vol. 55, no. 13, pp. 3032-3080, 2011.
[5] Bechler, M., Franz, W. J. and Wolf L. C. "Mobile Internet Access in Fleet Net," in Proc. of the 13th Fachtagung Kommunikation in Verteilten System (KiVS' 03), Germany, 2003, p. 107-113.
[6] Cabri, G., Leonardi, F. and Zambonelli x. "Engineering Mobile Agent Applications via Context-Dependent Coordination," IEEE Transactions on Software Engineering, vol. 28, no. 11, pp. 1039-1055, 2003.
[7] Camp, T., Boleng, J. and Davies V. "A survey of mobility models for ad hoc network research," Wireless Communication Mobile Computing, vol. 2, no. 5, pp. 483-502, 2002.
[8] Cao, M., Ma, W., Zhang, Q., Wang X. and Zhu W. "Modeling and performance analysis of the distributed scheduler in IEEE 802.16 Mesh Mode," in Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2005, p. 78–89.
[9] Han, Q., Bai, Y., Gong, L. and Wu W. "Link Availability Prediction-Based Reliable Routing for Mobile Ad Hoc Networks," IET Communications, vol. 5, no.16, pp. 2291-2300, 2011.
[10] Chuanlai Lu. "Queue Theory," Beijing University of Posts and Telecommunications Press, 1993.
[11] David, A., Maltz, Josh Broch and David B. Johnson. "Lessons from a Full-Scale Multi hop Wireless Ad Hoc Network Test bed," IEEE Personal Communications, vol. 8, no. 1, pp. 8-15, 2001.
[12] Deering, S. and Hinden R. "Internet Protocol version6 (IPv6) specifications," IETF RFC 2460, 1998.
[13] Haas, Z. J., Deng, J., Liang, B., Papadimitratos, P. and Sajama S. "Encyclopedia of Telecommunications Chapter-3," Wireless Ad Hoc Networks, John Wiley, 2002, p. 45-53.
[14] Han, Y., La, R. J., Makowski, A. M. and Lee S. "Distribution of path durations in mobile ad-hoc networks," Computer Networks, vol. 50, no. 12, pp. 1887-1900, 2006.
[15] Hayun Roy Ben "Java ME on Symbian OS: Inside the Smartphone Model," 1st Edition, Wiley, 2009.
[16] Kai Chen and Klara Nahrstedt. "iPass: An Incentive compatible Auction Scheme to Enable Packet Forwarding Service in MANET," in Proc. of 24th International Conference on Distributed Computing System (ICDCS'04), 2004, p. 534-542.
[17] Kaixin, Xu., Mario, Gerla., Lantao, Qi. and Yantai Shu. "TCP Unfairness in Ad Hoc Wireless Networks and a Neighborhood RED Solution," Wireless Networks, vol. 11, no. 4, pp. 383-399, 2005.
[18] Kumar, M., Shirazi, B., Das, S. K., Singhal, M., Sung, B. and Levine D. "Pervasive Information Communities Organization PICO: A Middleware Framework for Pervasive Computing," IEEE Pervasive Computing, vol. 2, no. 3, pp. 72-79, 2003.

[19] Lee, S. J., Gerla, M. and Chiang C. C. “On-demand multicast routing protocol,” in Proc. IEEE Wireless Communications and Networking Conference (WCNC ’99), vol. 3, 1999, p. 1298–1304.
[20] Li, B. and Wang K. H. “Non Stop: Continuous multimedia streaming in wireless ad hoc networks with node mobility,” IEEE J. Sel. Areas in Communications, vol. 21, no. 10, pp. 1627-1641, 2003.
[21] Luciano Bononi, Marco Conti and Lorenzo Donatiello, “Design and Performance Evaluation of a Distributed Contention Control Mechanism for IEEE 802.11 Wireless Local Area Networks,” ACM workshop on Wireless Mobile Multimedia, 1998, p. 59-67.
[22] Luo, H., Lu, S. and Bharghavan V. “A new model for packet scheduling in multi-hop wireless networks,” in Proc. 6th Annual International Conference on Mobile Computing and Networking (ACM MOBICOM ’00), 2000, p. 76–86.
[23] Michiardi, P. and Molva R. “CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks,” in Proc. of the 6th Joint Working Conference on Communications and Multimedia Security, vol. 100, 2002, p. 107-121.
[24] Nasim Faruque M., Ahmad, S. N. and Manoj Kumar. “Performance of QoS in Wireless Ad hoc Network for AODV Protocol using Fuzzy Based Techniques,” IJECT, vol. 2, no. 2, pp. 41-45, 2011.
[25] Perkins, C. E., Belding-Royer, E. and Das S. R. “Ad-hoc on Demand Distance Vector (AODV) Routing,” IETF Internet Draft, 2003.
[26] Perkins C. E. “Mobile IP Design Principles and Practices (Ed-2),” New Jersey: Addison Wesley Publications USA, 1998.
[27] Shor, J. and Robertazzi T.G. “Traffic sensitive algorithms and performance measures for the generation of self-organizing radio network schedules,” IEEE Transaction on Communications, vol. 41, no. 1, pp. 16-21, 1993.
[28] Sau and Scholefied C. “Scheduling for GPRS service class,” in Proc. of IEEE Wireless Communication and Networking Conference, vol. 3, 1999, p. 1229-1233.