Design and Performance Evaluation of Meghadoot – A Hybrid Wireless Network Architecture[†]

B. Venkata Ramana, Devesh Agrawal, and C. Siva Ram Murthy
Department of Computer Science and Engineering
Indian Institute of Technology Madras 600036 India
{vramana, dvagr}@cse.iitm.ernet.in and murthy@iitm.ac.in

Abstract— **This paper presents the design and evaluation of Meghadoot – a hybrid wireless network architecture that provides improved services, such as better connectivity among the users, efficient routing, and access to the Internet. Meghadoot architecture combines the advantages of both single-hop and multi-hop wireless networks. Meghadoot is extremely useful (a) as an alternative to existing wired or wireless last mile solutions, (b) to establish a network in a residential complex or in a university campus. The Meghadoot architecture has been implemented and tested in both the NS2 simulator and in the Linux kernel. Both, experimental and simulation results show an improved performance for Meghadoot, in terms of reduction in control overhead and increase in throughput over that of the existing approaches.**

I. Introduction

Over the past decade, the capabilities of wireless networks have increased manifold. These capabilities include, nomadic access, wider network coverage, increased bandwidths, and multi-hop relaying. These networks can be divided into single-hop and multi-hop wireless networks. Single-hop networks (e.g., cellular networks) offer wider network coverage and efficient routing. However, they are constrained by a high setup and maintenance costs and a high setup time. In addition, the available bandwidths are limited and expensive. Multi-hop networks (e.g., ad hoc wireless networks), on the other hand, are capable of operating without the support of any fixed infrastructure and centralized administration, and provide higher data rates compared to that of cellular networks. These networks use multi-hop relaying for the transmissions between the nodes that are not within the radio range of each other, thus extending the network coverage. However, the lack of connectivity to the Internet restricts the deployment of these networks.

This paper discusses a hybrid wireless network architecture called Meghadoot, which combines the best features of both single-hop (infrastructure based) and multi-hop (infrastructure less) networks, by using a centralized routing scheme for efficient routing and multi-hop relaying for an extended coverage. In Meghadoot, a specially designated node called Infrastructure Node, which can typically be a workstation computer, provides the centralized routing and other services to the nodes in its k-hop region called control zone, and also handles the route requests originated beyond k-hops that are destined to a node either in k-hop region or in the Internet.

[†]This work was supported by Microsoft Research University Relations, India.

Meghadoot finds its applications in the following environments, where the nodes will be moving at a low mobility: (a) as a distributed multi-hop wireless last mile solution, which is an alternative to the existing wired or wireless last mile solutions, (b) campus wide wireless connectivity for colleges and universities, and (c) connectivity to residential complexes. The advantages of such a system are (i) low cost of setting up and maintaining of the network, (ii) minimal configuration requirement, (iii) license-free bandwidth for communication within the local community, (iv) an extended network coverage, and (v) shared ownership of the network. Meghadoot is especially useful as a distribution network in the rural communication, since remote rural villages, with a cluster of less than hundred houses, located about 40 to 50 kilometers from the nearest town, are considered the most difficult places to provide a communication infrastructure. The difficulties are primarily due to (i) high cost of long-haul access link, (ii) cost of maintaining the distribution network, (iii) unavailability of power sources, and (iv) low traffic volume to recover the costs.

The rest of the paper is organized as follows. Section II briefly discusses the related work. Section III presents the architecture and related issues of Meghadoot. Section IV provides the implementation details of Meghadoot. Section V presents the experimental and simulation results and analysis of the results. Finally, Section VI summarizes our work.

II. Related Work

The authors of [1], propose a scheme that uses Ad hoc On-demand Distance Vector (AODV) and MobileIP protocols, to provide the Internet connectivity to the nodes in an ad hoc wireless network. The Foreign Agent (FA) periodically floods its advertisements in the entire network; on hearing of which, an unregistered node registries with the FA and obtains the care-of-address. All the registered nodes maintain their registration alive by periodically unicasting the registration requests. When FA receives a Route Request (RReq) packet and it sends a Route Reply (RRep) packet if it finds the intended destination in the RReq packet does not exist in the network. If a node is unable to receive a RRep from any node other than FA RRep within a specified time interval, it assumes that the intended destination is in the Internet, and proceeds to use the route contained in the FA-RRep.

As the amount of control overhead due to flooding of advertisements is extremely high, the mechanism in [2], which is an extension of above work, restricts the flooding zone to maximum k-hops, where k is a predefine parameter. When a node outside the k-hop zone wishes to access the Internet, it floods FA-Solicitation message. On receipt of such message, a registered node unicasts the message to its FA. Then the FA unicasts a FA-Advt to the soliciting node, which enables the node to register with it. In both [1] and [2], as every RReq leads to a network wide flood of AODV RReq packets, the routing overhead is also quite high. Because of this, the proposed mechanism limits the scalability of the network. Also, a high route setup time is incurred, especially, for Internet destinations as the route returned in the FA-RRep is not used immediately.

Another work *SOHAN* [3] aims at providing Internet access for larger regions. However, it requires Forwarding Nodes (FNs), which are fixed, high power, and dual radio nodes that provide routing services to all their one hop nodes. A separate channel is used to exchange routing messages across the FNs and also with the access point which connects FNs to the Internet. The specialized dual radio channel used in the FNs necessiates the need for special hardware. Also, as the FNs are fixed and serve only their one-hop nodes, a large number of FNs are needed for effective network coverage, hence increasing the overall cost of setting up the network.

Although the Meghadoot architecture might seem reminiscent of several hybrid wireless networks, such as [4]-[8], this however is not the case. For a good survey of these networks, please refer [9]. These hybrid wireless networks combine cellular and ad hoc modes of communication. Meghadoot is fundamentally different from hybrid wireless networks as it does not employ any cellular networking technology, hence it requires no specialized hardware such as expensive base stations and cellular towers, no relay nodes, no long range transmission radios, and no separate control channels, thereby significantly decreasing its deployment cost.

III. MEGHADOOT ARCHITECTURE

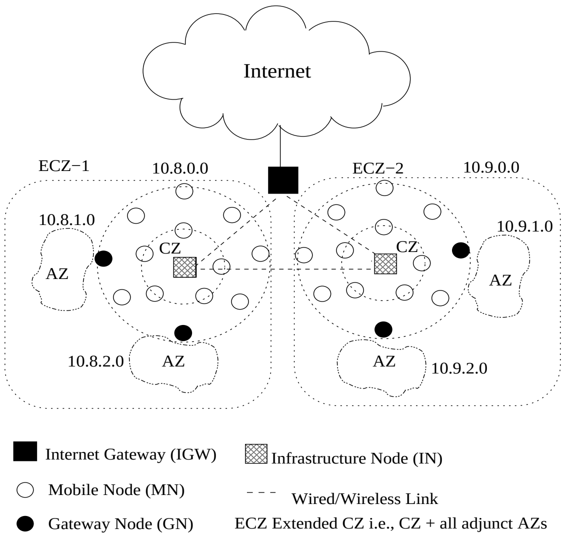

Fig. 1. An illustration of Meghadoot architecture.

Meghadoot is a hybrid wireless network architecture that is self organized and operates in a transparent manner to the users. It uses multi-hop wireless relaying to extend the network coverage as well as for forwarding data. The goal of Meghadoot is to provide an external connectivity to its users while minimizing the routing and other control overhead. The Meghadoot architecture consists of two different communication zones: Control Zone (CZ) and Ad hoc Zone (AZ). The operations in CZ, such as routing, registration, and other services, are controlled by a specially designated node called Infrastructure Node (IN). A CZ is formed by the Mobile Nodes (MNs) that are in the region of k-hop neighborhood of an IN and the MNs beyond k-hop neighborhood form one or more AZs. Apart from MNs and INs, Meghadoot consists of Gateway Nodes (GNs), which are MNs located at kth-hop from the IN. The GNs are part of both the CZ and the AZ and they hold the responsibility of interfacing MNs in the AZs to those in the CZ. Figure 1 shows an illustration of Meghadoot architecture. It contains two Extended Control Zones (ECZ)[1] Here, all the MNs within 2-hop ($k = 2$) neighborhood of IN form a CZ, and the remaining MNs form one or more AZs; the GNs provide limited routing services to the nodes in AZ.

A. Registration of MNs

The primary goal of IN is to control the routing, Internet access, and resource management in its k-hop neighborhood by maintaining information about the MNs in its CZ. For this purpose, the IN periodically floods advertisements (IN-Advt) in its k-hop neighborhood. These advertisements carry the information, such as address of the originating IN, radius of its CZ (i.e., k), and number of hops traversed so far (i.e., hop count). Through these advertisements, each MN in CZ learns its distance to the IN and next-hop node to reach the IN. An MN uses the hop count information to choose the nearest IN when it receives IN-Advts from more than one IN. On receipt of IN-Advt, an unregistered MN registers with the corresponding IN by sending a Registration Request (RG-Req) message to the next-hop node towards the IN. While forwarding a RG-Req packet, each MN on the path towards the IN, appends its address to the partial route contained in the RG-Req packet. Finally, when the IN receives a RG-Req packet, it unicasts a Registration Acknowledgment (RG-Ack) packet to the requesting MN along the reverse of the route contained in the RG-Req packet.

Each registered MN periodically refreshes its registration and also communicates its list of neighbors to the IN by means of a Neighbor Update (NU) message. We discuss the formation of the neighbor list later. The NU messages help the IN to keep track of the approximate topology of its CZ. In order to reduce the size of the NU message, an MN sends the incremental update of its neighbor list to the IN. Note that the MNs beyond the k-hop neighborhood do not get any IN-Advts and are said to be in the AZ.

However, the MNs in the AZ may also register with the IN in an on-demand manner as mentioned in [2]. In this case, the

[1]ECZ is the union of the CZ and all its adjacent AZs. All nodes in czSubnet belong to the ECZ.

MN broadcasts a solicitation message in its AZ, on receiving which the GN forwards it to its IN. The IN then unicasts an IN-Advt to this interested MN. The MN may then register with this IN in the manner outlined above. Note that registered nodes in the AZ do not periodically send NU messages to their respective IN's. However, they need to re-register with their INs before their registration expires.

The MNs can build their neighbor list by the following way. Each MN periodically broadcasts a beacon called MNBeacon in its one-hop neighborhood, by exchanging which, each MN gets to know its one hop neighbors. This list of neighbors is then communicated to the IN periodically in a NU packet, thus enabling the IN to construct the entire topology of its CZ. However, this periodic MNBeaconing increases the control overhead. In order to reduce this overhead, as the neighboring MNs can learn the presence of an MN by overhearing MN's transmissions, an MN sends an explicit MNBeacon only when it has not transmitted any data or control packets for the duration of the MNBeacon interval.

B. Addressing

In Meghadoot, every CZ is associated with a unique subnet address (*czSubnet*) for which the IN of the CZ is the ingress router. The IN assigns unique addresses to the MNs in its CZ at the time of their registration. For each of its adjacent AZ, the IN assigns a unique subnet address which is a part of its czSubnet and also keeps track of at least one GN, called *inchargeGN*. It stores the subnet address of each AZ (*azSubnet*) and its corresponding inchargeGN, in a table called *azTable*. Each inchargeGN periodically broadcasts its assigned subnet address, to enable MNs in its AZ to obtain a unique IP address. A scheme involving random address configuration and duplicate address detection similar to the one found in [1] is employed to assign unique addresses to AZ MNs. We employ a scheme similar to the MobileIP [10], to do address assignment for MNs that originally belong to one CZ and have moved to another CZ. Note that we only provide mobility freedom to CZ MNs. Each registered MN keeps track of the following information:

- *HIN*: The Home IN (HIN) or the IN of the CZ to which this MN originally belongs to, initially set to null.
- *curIN*: The IN of the new CZ this MN has moved into and seeks to register with.
- *mnAddr*: The current address of the MN, which was assigned to it by its HIN, initially set to null.

Each RG-Req sent by the MN contains the above information in the fields RG-Req.HIN, RG-Req.mnAddr, RG-Req.openTCPConn. The IN classifies its registered MNs into three types: *home* MNs are those that have registered with it and are currently in its CZ, *outside* MNs are those that have registered with it but are currently in some other CZ, and *foreign* MNs are those that originally belong to some other CZ but have moved into its CZ. The pseudo code for registering an MN at the IN is given below.

```
1:  if (RG-Req.mnAddr = null) OR (Re-Req.mnAddr does not
    belong to $czSubnet$ AND RG-Req.HIN = null) then
2:      /* Acquiring a new address */
3:      Let ipAddress = new IP Address belongs to $czSubnet$
4:      Register ipAddress as $home$ node
5:      Send ipAddress in RG-Ack to this MN
6:      Update the DNS tables
7:  else if (RG-Req.mnAddr does not belong to $czSubnet$)
    AND (RG-Req.HIN is not this IN) then
8:      This is a $foreign$ node,
9:      if (RG-Req.openTCPConn = false) then
10:         Inform HIN of this MN and upon getting ack
            from it, register this node as $foreign$ node.
11:     else
12:         Perform steps from 2 to 6 and inform old HIN
            of this MN about this address change.
13:     end if
14: else if (RG-Req.HIN is this IN but not the RG-Req.curIN)
    then
15:     An $outside$ MN that has come back to its HIN. So,
        re-register this node as home node, and forget its old
        IN, if any.
16: else
17:     This is a registered node, so refresh its registration.
18: end if
```

Note that each IN keeps track of current foreign IN for each of its *outside* MN, and the HINs for each of its *foreign* MNs.

C. Name Resolution Service

Domain Name Service (DNS) facility is provided only for the registered MNs in the Extended Control Zone (ECZ). Each MN has a unique name, that must be associated with its current address. Note that when an MN moves into a new CZ, its address may change as the MN is allowed to acquire a new address when it moves into a new CZ with no active TCP connections. A Meghadoot network is served by a master name-server, which devolves the name resolution of the constituent CZs to their respective IN's. During registration the MN configures its current IN as its primary name-server. The IN updates the current address of the MN in the Master name-server whenever the MN acquires a new address.

D. Routing Mechanism in Meghadoot

We employ a centralized source routing based protocol for routing in the CZ, which is called as *Meghadoot Routing Protocol (MRP)*. The optimizations mentioned in Section III-E reduce the network overhead attributed to routing. The IN maintains the approximate topology of its CZ as mentioned in III-A and handles the routing for MNs in its ECZ. Each inchargeGN is responsible for forwarding data across its AZ to the CZ it is a part of. Route discovery can be understood by considering the following cases:

1. Source and destination belong to the same ECZ: This case can be further decomposed into four cases, each of which is illustrated in Fig. 2:

a) **Both source and destination are in the same CZ:** Since the IN maintains the topology of its CZ, an MN only needs to ask the IN for a route to the destination MN. Hence, it unicasts (not broadcasts) a RReq to the IN. The IN then replies with a

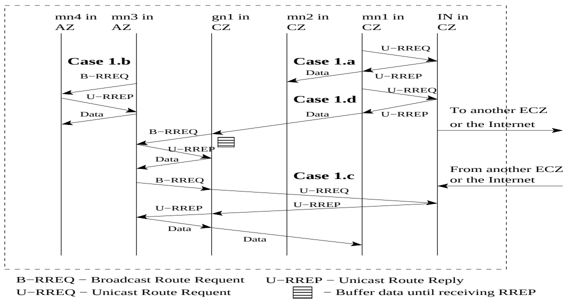
mn4 in AZ
mn3 in AZ
gn1 in CZ
mn2 in CZ
mn1 in CZ
IN in CZ
Case 1.a
Case 1.b
Case 1.c
Case 1.d
U−RREQ
U−RREP
B−RREQ
Data
To another ECZ or the Internet
From another ECZ or the Internet
B−RREQ − Broadcast Route Requent
U−RREP − Unicast Route Reply
U−RREQ − Unicast Route Requent
▤ − Buffer data until receiving RREP

Fig. 2. Routing among MNs in the Extended CZ.

corresponding RRep if it is able to compute a route from the source MN to the destination MN.

b) **Both source and destination are in the same AZ:** There is no topology information of the AZ since there is no IN associated with it. Hence, we need to employ a pure ad hoc routing protocol in the AZ. Essentially the source floods the RReq, to which the destination replies with a unicasted RRep to the source.

c) **Source is in the AZ and the destination is in the CZ:** In this case the GN must help to transfer the RReq from the AZ to the CZ. When a flooded AZ RReq reaches a GN, the GN unicasts it to its IN if the destination does not belong to its $azSubnet$. The IN, then computes a route from the GN to the destination and returns the requesting MN (in the AZ) a route from the source to the GN to the destination. Thus GNs help to route traffic across the CZ and the AZ.

d) **Source is in the CZ and the destination is in the AZ:** Similar to the previous case, the GN helps in transferring RReq from the CZ to the AZ. On getting a RReq, the IN determines the AZ in which the destination lies and finds the inchargeGN for that AZ. A RRep containing the route from the source to the inchargeGN is sent. The packets are source routed to the inchargeGN, which on receiving them does an AZ route discovery for the destination if required, buffering the data packets in the meanwhile.

2. Source and the destination belong to different ECZ: This process can be conceptually viewed as finding the route from the source to its IN, from the source IN to the destination IN (i.e., the IN in whose ECZ destination lies), and from the destination IN to the destination. Note that routing from one IN to another is taken care of by the routing protocol running on the backbone network (e.g., OSPF, RIP). We can find a route from a node to its IN by setting the destination to be the IN in case 1. Since each node must be a part of some ECZ, a route between any two nodes can be found from the above cases.

E. Optimizations

In order to decrease the routing overhead, the following optimizations have been incorporated. Note that most of these optimizations are similar to the ones mentioned in [11].

1) *LinkGraph*: We use a link cache as mentioned in *Appendix A* of [11] to enhance an MN's knowledge of its local topology.
2) *NU aggregation*: An MN can piggyback its own NU message as it forwards NU messages of other MNs.
3) *Caching overheard routes*: An MN caches the routes it overhears in the source routed data packets.

IV. Prototype Implementation of Meghadoot Architecture

The Meghadoot architecture has been implemented as a loadable Linux kernel module (LKM). We use the netfilter framework [12] to register handlers for each netfilter hook. The netfilter hooks allow us to capture packets from both the IP and the MAC layers. These message handlers implement the entire functionality of the Meghadoot architecture. Kernel timers [13] are used for bookkeeping and for periodic tasks. The current implementation adds a custom header to all packets being transmitted in the Meghadoot network. Each MN was put into promiscuous mode for better neighbor discovery as we could not find a way to know of the RTS/CTS exchanges by our 802.11 interface. For further details on the Meghadoot prototype implementation, interested readers can refer to [14].

V. Experimental and Simulation results

A. Experimental Results

Our experimental setup consisted of two Linux PCs and four laptops each fitted with *US Robotics (USR012415)* 802.11 cards with *prism* chip sets. We measured the TCP throughput in the CZ and AZ for varying hop lengths. The throughput is measured by transferring files using FTP. We also have a script running to randomly *ping* known and unknown destinations, in order to generate a moderate background routing overhead. As our experimental setup had only six nodes, we could not test our prototype more extensively.

Figure 6 shows the experimental results of average TCP throughput in CZ and AZ for different hop lengths between the source and destination MNs. The average TCP throughput is obtained by averaging the individual throughput of several runs of an FTP session that transmits files of different sizes. Figures 3, 4, and 5, show variations in throughput with respect to file sizes for different hop lengths. In these figures, we observe that the throughput in CZ is significantly higher than that of AZ for increasing hop lengths; also as expected the throughput decreases with increasing hop lengths for both CZ and AZ flows. The reason for better performance in CZ as compared to AZ is that the $RReq$'s generated by the $ping$ script are broadcasted in case of AZ as compared to in CZ where they are unicasted to the IN. The higher routing overhead in the AZ increases contention and hence the TCP throughput suffers. This effect is more pronounced as the hop length between the source and the destination is increased.

B. Simulation Results

We have carried out extensive simulations using ns-2.28 [15] for measuring the performance Meghadoot. As MRP is the core part of Meghadoot, we evaluate MRP and compared our results with that of Dynamic Source Routing (DSR) protocol. The various parameters used in our simulation are listed in Table I. The AZ performance evaluation is not required, as any existing ad hoc routing scheme could be

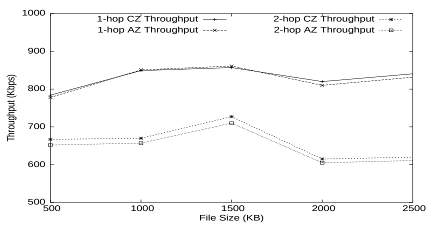

Fig. 3. Throughput vs File Sizes for Hop Lengths 1 & 2.

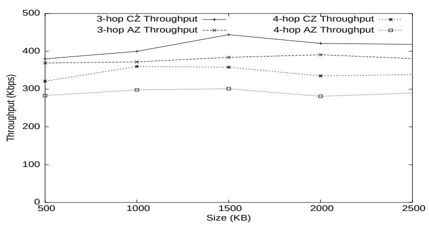

Fig. 4. Throughput vs File Sizes for Hop Lengths 3 & 4.

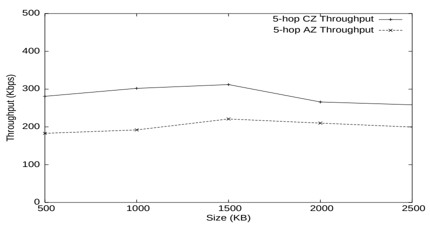

Fig. 5. Throughput vs File Sizes for Hop Length 5.

TABLE I

SIMULATION PARAMETERS USED IN ns-2 SIMULATION FRAMEWORK

Description	Value
Simulation area/Node placement	1200m×1200m/Random placement
Number of INs/Number of MNs	1 IN / 74 CZ MNs
Location of the IN	Center of the simulation area
Transmission range of each MN	250m
MAC/Application protocol	802.11b with 2Mbps/CBR
CBR traffic rate/Packet size	5 packet per second/512 bytes
Mobility model	Random way-point with constant Mobility of 1m/s and zero pause time
IN-Advt/MNBeacon/NU interval	10/8/10 in seconds
Number of different seeds	25 seeds with 95%confidence level

employed in it. Hence, we evaluate the performance of only CZ. Unless otherwise stated, we use the same setup for all our simulation studies. The metrics used for comparison are Packet Delivery ratio (PDR), Signaling Overhead (SO)[2].

Several simulation studies are carried out to evaluate the performance of MRP versus that of DSR. The first two studies measure the variation in the metrics with increasing traffic, the third one measures the variation with increasing mobility, and finally, the fourth one measures the variations with increasing CZ radius. We consider two classes of traffic patterns. Since Meghadoot was primarily designed for Internet connectivity, the first traffic pattern class, henceforth referred to as the *Internet traffic pattern*, has 80% of all flows destined to the Internet and the remaining 20% being between randomly chosen source and destination pairs. Since Internet traffic has to go through the IN, we approximate Internet traffic with traffic to the IN. The second traffic pattern class, henceforth, is referred as the *Random traffic pattern*.

The MRP control overhead can be divided into two parts. The fixed part concerns the proactive maintenance of the updated topology at the IN, whereas the variable part consists of the route request-reply exchanges. The variable part increases with increasing number of flows, number of nodes, and mobility. However, it increases only slightly as the RReqs are unicasted to the IN. On the other hand, the DSR control overhead increases at a much larger rate with increasing number of flows and mobility, because the RReqs are flooded in it. Due to this reason, MRP achieves a better PDR than DSR with a much lesser overhead.

Internet traffic pattern: In this study, we evaluate the performance of MRP versus DSR, under the *Internet traffic pattern class*, when the number of flows is increased from 10 to 16. Figs. 7 and 8 illustrate the PDR and the SO, respectively. We observe from Fig. 8 that the SO of MRP is much lower than that of DSR. The high control overhead leads to an increased contention, particularly around the IN, hence the PDR of DSR falls sharply with increasing number of flows. Note that Fig. 8 might seem contradictory to our claim that the SO increases with increasing number of flows. It can be explained as follows. Since a large fraction of the traffic is towards the IN, nodes along the paths to the IN cache the route to it; hence, obviating the need for a RReq to be done by these nodes. Since RReqs are flooded in the case of the DSR, we see that the SO of DSR (refer Fig. 8) falls even sharply with increasing number of flows.

Random traffic pattern: In this study, we evaluate the performance of MRP versus DSR, under the *Random traffic pattern class*, when the number of flows is increased from 10 to 16. Figs. 9 and 10 illustrate the PDR and the SO, respectively. Similar to the previous study, the SO of MRP is much lower than that of DSR, hence leading to a higher PDR because of the reduced contention. Note that unlike the previous study, the SO does not decrease with increasing number of flows, this is because the likelihood of caching decreases when most of the flows are to different destinations. The PDR falls with increasing number of flows due to the increased contention caused by more traffic.

Evaluation with increasing mobility: Here, we evaluate the performance of MRP versus DSR with increasing mobility under the *Internet traffic pattern class*. The number of flows is fixed as 10 and the mobility is varied from 1 m/s to 5 m/s. Note that, as most of the traffic in rural settings is pedestrian, we specifically tuned Meghadoot for low mobility scenarios. It can be observed from Fig. 11 that MRP achieves almost the same PDR as DSR. Fig. 12 shows that MRP has a lesser SO than that of DSR. The sudden increase in the SO of MRP at around 3 m/s is because the beaconing rates of MRP (given in Table I), work well till about 3 m/s, after which more link-breaks start happening, leading to an increased control overhead. Note that the SO of DSR rises even more sharply with increasing mobility as each link-break triggers a RReq flood.

Evaluation with increasing CZ radius: In this study, we evaluate the performance of MRP versus DSR with increasing CZ radius under the *Random traffic pattern class*. The number of flows is fixed as 10, and the CZ radius is varied from 3 to 6. The number of nodes increases along with the CZ radius. We observe from Fig. 13 that the SO of DSR rises at a much higher rate than MRP. The DSR's SO is primarily due to the flooding of RReqs, which is proportional to the number of nodes and to

[2]PDR = The total number of data packets received by all the destinations over the total number of data packets sent by all the sources and SO = The ratio of the total number of control packets exchanged in the network over the total number of data packets exchanged.

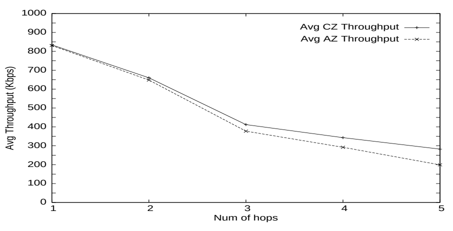

Fig. 6. Average TCP Throughput in CZ and AZ vs Hop Length.

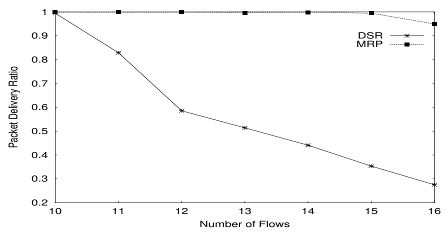

Fig. 7. Internet traffic pattern: PDR vs Number of flows.

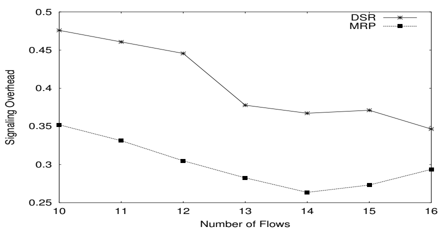

Fig. 8. Internet traffic pattern: SO vs Number of flows.

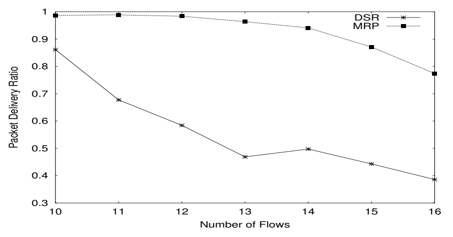

Fig. 9. Random traffic pattern: PDR vs Number of flows.

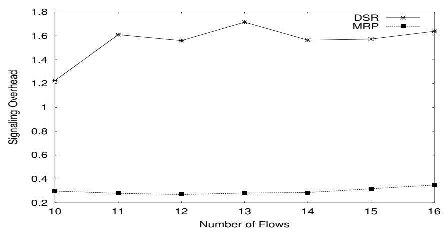

Fig. 10. Random traffic pattern: SO vs Number of flows.

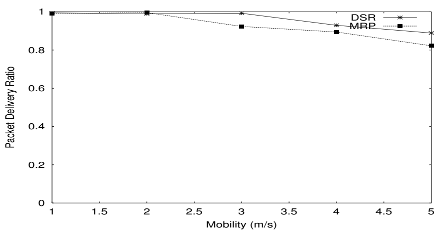

Fig. 11. Internet traffic pattern: PDR vs Mobility.

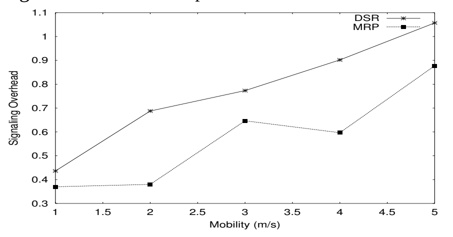

Fig. 12. Internet traffic pattern: SO vs Mobility.

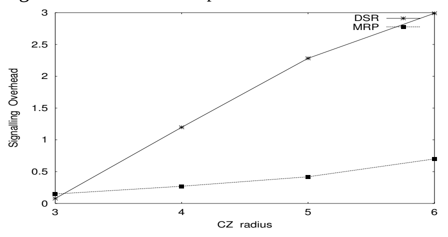

Fig. 13. Random traffic pattern: SO vs CZ radius.

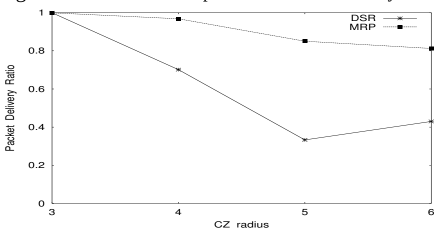

Fig. 14. Random traffic pattern: PDR vs CZ radius.

the number of flows. Whereas, the MRP's SO is proportional only to the number of nodes in the network. Hence, for a given number of flows and with increasing number of nodes, the overhead of DSR rises at a greater rate than that of MRP. Therefore, we notice in Fig. 14, that the higher rate of increase of the DSR's SO leads to a sharper fall in the DSR's PDR, as compared to MRP, with the increasing CZ radius. Similar behavior is also observed for the *Internet traffic pattern*.

VI. Summary

In this paper, we described the design Meghadoot – a hybrid wireless network architecture and evaluated the routing protocol (MRP) used in it by means of simulation and experimental testing. These results confirm that Meghadoot performs better, in terms of higher packet delivery ratio and lower signaling overhead over that of the existing approaches. Meghadoot is especially suited to be an alternative to existing last mile solutions for establishing communication network among rural communities, and to set up low cost minimum configuration Internet connected networks in residential areas and university campuses.

Our future work consists of studying the performance of Meghadoot against the one in [2] which proposes a mechanism to provide the Internet connectivity to the users of ad hoc wireless networks, and a hierarchical routing protocol, such as Optimal Link State Routing (OLSR) protocol. We shall also look into the aspects of load balancing across control zones, security, and resource management in Meghadoot.

References

[1] Y. Sun, M.B. Elizabeth, and C. E. Perkins, "Internet Connectivity for Ad hoc Mobile Networks," *Intl. Journal of Wireless Information Networks*, vol. 9, no. 2, pp. 75-88, Apr 2002.
[2] P. Ratanchandani and R. Kravets, "A Hybrid Approach to Internet Connectivity for Mobile Ad Hoc Networks," *in Proc. IEEE WCNC*, vol. 3, pp. 1522-1527, Mar 2003.
[3] S. Ganu, S. Zhao, L. Raju, B. Anepu, I. Seskar, and D. Raychaudhuri, "Architecture and Prototyping of an 802.11-based Self-organizing Hierarchical Ad Hoc Wireless Network (SOHAN)," *in Proc. IEEE PIMRC*, vol. 2, pp. 880-884, Sep 2004.
[4] H. Luo, R. Ramjee, P. Sinha, Li (Erran) Li, S. Lu, "UCAN: A Unified Cellular and Ad-hoc Network Architecture," *in Proc. ACM Mobicom*, pp. 3.0-367, Sep 2003.
[5] H. Wu, C. Qiao, S. De, and O. Tonguz, "Integrated Cellular and Ad hoc Relaying Systems: iCAR," *in IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2105-2115, Oct 2001.
[6] H. Hsieh and R. Sivakumar, "On Using Peer-to-peer Communication in Cellular Wireless Data Networks," *in IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 57-72, Jan 2004.
[7] H. Hsieh and R. Sivakumar, "On Using the Ad-hoc Network Model in Cellular Packet Data Networks," *in Proc. MobiHoc*, pp. 36-47, Jun 2002.
[8] H. Hsieh and R. Sivakumar, "A Hybrid Network Model for Wireless Packet Data Networks," *in Proc. Globecom*, pp. 961-966, Nov 2002.
[9] C. Siva Ram Murthy and B. S. Manoj *Ad Hoc Wireless Networks, Architectures and Protocols*, Prentice Hall, New Jersey, 2004.
[10] C. Perkins, "IP Mobility Support," *RFC 2002*, Oct 1996.
[11] D. B. Johnson, D. A. Maltz, and Y. Hu, "The Dynamic Source Routing Protocol for Mobile Ad hoc Networks," *Internet Draft Version 10*, 2004.
[12] R. Russell and H. Welt, "Linux netfilter Hacking HOW-TO," *http://www.netfilter.org*, Jul 2002.
[13] Jonathan Corbet, Alessandro Rubini, and Greg Kroah Hartman, *Linux Device Drivers*, O'Reilly Publishers, 2005.
[14] V. Kumar, "A Kernel Implementation of the Meghadoot Architecture," *Tech Report*, Indian Institute of Technology Madras, May 2005.
[15] The Network Simulator - NS2 *http://www.isi.edu/nsnam/ns*.