MIDTONE: Multicast in Delay Tolerant Networks

Zunnun Narmawala
Institute of Technology
Nirma University
Ahmedabad 382 481, India
Email: zunnun.narmawala@nirmauni.ac.in.

Sanjay Srivastava
Dhirubhai Ambani Institute of
Information and Communication Technology
Gandhinagar 382 007, India
Email: sanjay_srivastava@daiict.ac.in.

***Abstract*—Delay Tolerant Networks (DTN) are sparse ad hoc networks in which no contemporaneous path exists between any two nodes in the network most of the time. Due to non-availability of end-to-end paths, multicast protocols of traditional networks fail in DTN because they try to find connected multicast graph between source and destination nodes before forwarding data packets. Routing protocols proposed for DTN follow 'store-carry-forward' paradigm in which two nodes exchange messages with each other only when they come into contact. In the process, 'Single-copy' schemes maintain only one copy of the message in the network at any time and the forwarding node waits for the pre-determined next node to transfer the message. 'Multi-copy' schemes spread more than one copy of the message opportunistically when nodes come into contact rather than waiting for pre-determined next node. While Multi-copy schemes improve chances of delivery and work well even without any knowledge of the network, communication overhead and buffer occupancy are quite high for these schemes. We propose Multi-copy routing protocol for multicasting in DTN called "Multicast In Delay TOlerant NEtworks (MIDTONE)" which uses 'Network coding' to reduce this overhead without compromising performance. Network coding is a mechanism in which nodes encode two or more incoming packets and forward encoded packets instead of forwarding them as it is. We also propose three novel packet purging schemes to drain packets out of the network which takes advantage of features of network coding to increase buffer efficiency. As simulation results suggest, our protocol achieves significantly less delay to deliver all the packets in infinite buffer case and higher delivery ratio in finite buffer case compared to non-network coding based Multi-copy scheme. We also provide empirical relation to estimate optimal generation size for given network and performance parameters.**

I. Introduction

Delay Tolerant Networks (DTN) are ad hoc networks in which end-to-end connectivity between source and destination may never be present. Such environments can be found, for example, in very sparse mobile ad hoc networks where nodes 'meet' only occasionally to be able to exchange information, or in wireless sensor networks where nodes sleep most of the time to conserve energy. Conventional routing protocols as well as routing protocols for mobile ad hoc networks such as DSR, AODV etc. assume that complete path exists between source and destination and try to discover the path before any useful data is sent. i.e., if no end-to-end path exists most of the time, these protocols fail to deliver any data to all but the few connected nodes. Many DTN applications need multicast service. For example, in military battle fields, it is necessary to transmit orders from a command center to a group of field commanders [1]. However, traditional multicast methods proposed for the Internet or mobile ad hoc networks [2]–[5] are not suitable for DTN, due to the challenge of frequent network partitions. Data transmissions suffer from large end-to-end delays along the tree because of the repeated partitions due to frequent disconnections. Also the traditional approaches may fail to deliver a message when the possibility of link unavailability becomes high.

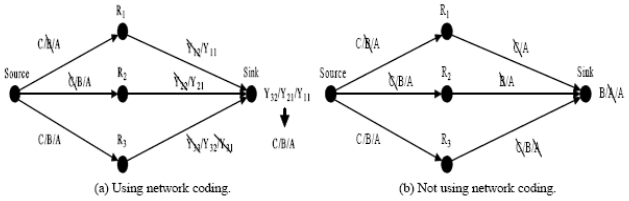

Fig. 1. Example illustrating benefits of network coding in unreliable networks (taken from [6])

To increase chance of delivery and to reduce delivery delay, routing approaches in DTN make multiple copies of a packet in the network. However communication overhead and buffer occupancy increases as we increase number of copies per packet. Network coding can reduce this overhead without compromising on performance. In network coding, instead of forwarding packets as it is, nodes may recombine two or more input packets into one or more output packets. The successful reception of information does not depend on receiving specific packets but on receiving sufficient number of independent packets. It is illustrated by the following intuitive example (taken from [6]) shown in Fig. 1. In the example, packets A, B and C are to be sent from source to sink using multipath as the links are lossy. In Fig. 1(a), routers R1, R2 and R3 encode incoming packets into same number of outgoing packets while in Fig. 1(b), packets are transmitted as it is. In the second case, as two packets are redundant, only packets A and B are received successfully while in the first case, with high probability, all the three packets A, B and C will be received successfully. i.e., network coding is more effective in delivering packets successfully. So, similar performance can be achieved in network coding based scheme compared to non-network coding based scheme while making less number of copies per packet.

Further, multiple copies of a packet increase buffer require-

ment of nodes in the network. For multicasting, copies of the packet need to be kept in buffer for longer duration because packet needs to be delivered to multiple destinations. So, efficient buffer management is very crucial for multicasting in DTN. Network coding can use limited buffer more efficiently because instead of dropping packets, it can reduce number of packets by combining existing packets.

The rest of the paper is structured as follows: In section II, we present existing work and identify open issues. We describe our protocol MIDTONE in section III and IV. Section V presents simulation results. Finally, in section VI, we present our conclusions.

II. Related Work

Ahlswede et al. in their pioneering work [7] showed that maximal network flow and sum of cut-edge capacities of any minimal cut that separates sender and receiver are equal and it is the maximum achievable capacity. This capacity is generally not achievable in conventional communication network which considers information as a 'fluid' which can simply be routed or replicated. i.e., independent information streams may share network resources but they are kept separate. In network coding, instead of forwarding packets as it is, nodes may recombine two or more input packets into one or more output packets. Li et al. in [8] showed that linear coding with finite field size suffices to achieve max-flow from the source to each destination. Among various linear encoding techniques at network nodes, Random Linear Coding (RLC) [9] is widely used as it is distributed and less computationally intensive and algorithms are well understood. Further, probability of receiving dependent packets is very low.

We focus on Multi-copy schemes as they are more robust and provide acceptable delivery delay. Further, Multi-copy schemes generally do not use any prior information of the network which is generally not available in DTN. As the main disadvantage of such schemes is overhead, focus of such schemes is to reduce overhead.

Various Multi-copy schemes to unicast in DTN are suggested in literature. Vahdat and Becker presented a routing scheme called as 'Epidemic Routing' [10]. In this scheme, nodes exchange packets opportunistically whenever they come into contact with any other node. i.e., packets are flooded in the network. So, if network resources are unlimited, the scheme will give optimal performance. But, network resources such as buffer space and bandwidth are generally limited. To limit the number of copies per packet, Spyropoulos et al. in [11] proposed 'Spray and Wait' scheme which spreads only limited number of copies per packet in the network with optimal binary spreading mechanism. Once given number of copies of a packet are spread, each node having a copy of the packet waits till it comes into contact with destination node. Copies of a packet should be purged from network nodes once the packet is delivered for efficient buffer and bandwidth utilization. Small and Hass in [12] proposed 'VACCINE' recovery scheme in which destination node generates an 'anti-packet' for each received packet. Nodes accumulate information of packets delivered at destinations by receiving anti-packets from destination nodes and by exchanging accumulated information with other nodes. Using this information, they drop copies of the delivered packets from their buffers.

Zhang et al. in [13] investigated the benefits of using RLC for unicasting in DTN. They used Spray and Wait type of forwarding mechanism for forwarding encoded packets. Packets are grouped into 'generation' which is defined as group of packets which can be mixed in order to generate encoded packets in RLC scheme. i.e., any node can generate encoded packets only from packets of same generation. They also presented modified VACCINE recovery scheme which purges generations from the network instead of individual packets. i.e, an anti-packet is sent for a generation whenever the whole generation is received at a destination.

In DTN, as group membership may change during multicast packet transfer due to large delays, it is required to define intended receivers of the multicast packet unambiguously from the group membership. Zhao et al. in [14] have proposed different group membership models and routing schemes to multicast in DTN. Among these schemes, Broadcast-Based Routing (BBR) is the only Multi-copy scheme which simply floods packets in the network. It gives the best performance when no network information is used but with significant overhead.

No network coding based Multi-copy scheme for multicasting in DTN is suggested. Benefits of network coding based multicast in traditional networks are well understood [7], [15]–[20]. So, we propose a Multi-copy protocol to multicast in DTN with network coding. In this scheme, draining packets out of the system (purging) once they are delivered is a challenging issue as there are more than one receivers of a generation and each generation has more than one packet. We propose three novel purging schemes to purge packets out of the system. Moreover, size of the generation has significant impact on delivery delay and efficiency. In the present work, no attention is paid to tune generation size for getting desired performance. We analyze effect of generation size on network and performance parameters.

III. Overview of the Protocol: MIDTONE

Following are the main approaches used in our protocol.

A. Spray and Wait Forwarding

To reduce resource usage, number of copies per packet are controlled by using Spray and Wait mechanism [11] in which only specified number of copies (L) are spread in the network.

B. Network Coding

In existing routing schemes for DTN, whenever a transmission opportunity arises, ideally, a node should forward packets such that the destination node gets all the required packets without getting any redundant packet. However, a forwarding node has no such precise knowledge in DTN. Therefore, it is difficult to select the best packet for transmission. On the other hand, in the network coding based protocol, a node

can transmit any of the coded packets since all of them can contribute the same to the eventual delivery of all data packets to the destination with high probability. Similarly, the network coding based protocol has the advantage in utilizing limited buffer resource since dropping any coded packet has the same effect. So, our protocol uses network coding to exploit these and other benefits mentioned earlier. As our focus is not to investigate better coding schemes and RLC provides reasonably good performance, we use RLC in our protocol.

C. Multicasting

Our protocol can run on top of the group membership models of multicast in DTN [14] and it will not hamper the performance of the protocol as the protocol's forwarding policy is independent of who the intended destinations are. Rather, in the situations where intended destinations change frequently, the proposed protocol has an edge over the schemes which try to find routes to destinations because such schemes will need frequent updation of the routes while the proposed protocol will not require such updations.

D. Purging Schemes

We propose following purging schemes for network coding based multicast in Delay Tolerant Networks. These schemes have two parts: 1) **Delivery Information Propagation:** It explains how the information about delivery of packets or generations at destination nodes is propagated in the network and 2) **Actual Purging:** It explains how nodes purge packets or generations from their buffers upon receiving delivery information.

1) Probabilistic Purging (PBP):

- *Delivery Information Propagation:* In this scheme, whenever a destination node receives entire generation, it immediately sends an anti-packet to the neighbour node from which it has received last independent packet of the generation. This anti-packet contains identifications of newly received generation and the receiver node. Each node maintains, for different generations, list of destinations which have received the generation. Please note that this list is accumulated at each node by receiving anti-packets from destination nodes and exchanging the list with other nodes.
- *Actual Purging:* When two nodes come into contact, they exchange accumulated lists. For each destination of a generation listed in the received list which is not in the node's list, entire generation is purged with some probability. The idea is to decrease number of nodes having encoded packets of the generation as more and more destinations receive the generation. Whenever the list of destinations (which have received a generation) at a node contains all the destinations of a generation, the entire generation is purged from node's buffer if it has not already been purged probabilistically.

2) Proportional Purging (PPP):

- *Delivery Information Propagation:* It is same as PBP.
- *Actual Purging:* When two nodes come into contact, they exchange accumulated lists as in PBP. For each destination of a generation listed in the received list which is not in the node's list, $rank$ of the generation is decreased proportionally where $rank$ denotes number of independent packets of a generation stored in a buffer. The idea is to purge number of independent packets from forwarding node's buffer proportional to number of destinations which have received the generation. Whenever the list of destinations (which have received a generation) at a node contains all the destinations of a generation, the entire generation is purged from that node's buffer if the node has any encoded packets stored in its buffer for that generation.

3) Aggressive Purging (AP):

- *Delivery Information Propagation:* In this scheme, each destination node sends a new anti-packet for each new independent packet of a generation received. This anti-packet contains unique identification of the generation and $rank$ of the generation along with identification of the receiver node. The nodes in the network accumulates information about $rank$ of generations at different destinations by exchanging this information with each other. i.e, it compares the $rank$ of a generation at a destination received from the neighbour with the $rank$ for the same stored in it. If the received $rank$ is greater, then it updates the $rank$ stored in it. Please note that the overhead of this scheme is high compared to other two schemes because a destination node sends an anti-packet for each new independent packet received and the information to be exchanged between two nodes when they come into contact also contains values of $rank$ at all destinations along with their identification.
- *Actual Purging:* Upon receiving an anti-packet, the forwarding node decreases number of independent packets of a generation stored in its buffer proportional to the total number of independent packets received at all destinations. The idea is to reduce number of independent packets stored in the network as they are delivered at destinations. Similar to PPP, whenever total of values of $rank$ at all destinations for a generation accumulated at some node is equal to $G * TotalDest$ where G denotes size of the generation and $TotalDest$ denotes total number of destinations, the entire generation is purged from that node's buffer if it has any encoded packets stored in its buffer for that generation.

IV. Protocol Description

In this section, we describe working of our protocol. Data packets are grouped into generations. Nodes store independent packets along with their coefficients according to RLC scheme. We define a new variable $token = L * G$ which denotes number of encoded packets in the network for a generation.

Main components of our protocol are depicted in Fig. 2 and are explained in following sections.

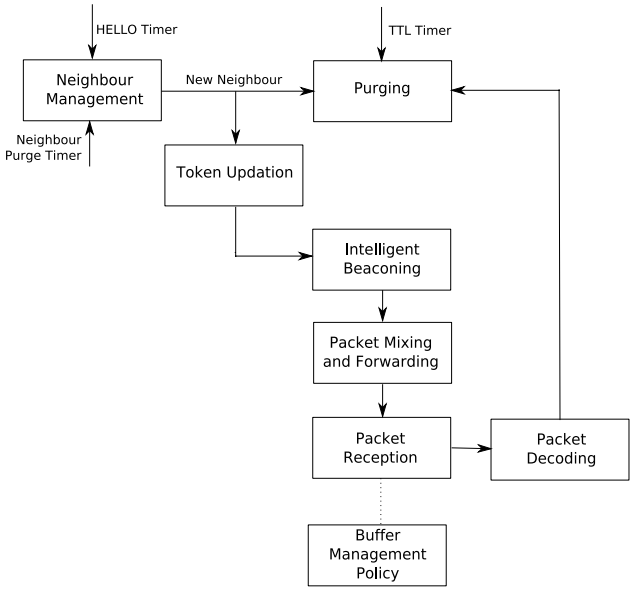

Fig. 2. Schematic diagram depicting main components of the protocol

A. Neighbour Management

Every node in the network transmits 'HELLO' packet at fixed interval called as 'HELLO Timer'. Duration of the interval is pre-configured parameter which is decided depending on speed of the nodes and their communication range. e.g., if we consider the case of neighbour nodes moving with 6 m/s speed in opposite directions having communication range of 100 m, they will remain in contact for approximately 34 seconds. If we keep the HELLO Timer value to be 10 seconds, the neighbour node will be detected at most after 10 seconds from the actual time of first contact between two nodes and the node will not be able to make use of approximately $1/3^{rd}$ of total contact period. If we decrease HELLO Timer value, then number of HELLO packets in the network will increase but neighbour will be detected early. So there is a tradeoff.

Whenever a node receives HELLO packet from another node, it adds that node into its neighbour list if that node is not in the list already. If any node does not receive any HELLO packet for the interval equal to three times HELLO Timer from its neighbour node, that node is purged from the neighbour list. This interval is called as 'Neighbour Purge Timer'.

B. Purging

After adding a node into the neighbour list, the node shares accumulated delivery information with its neighbour for purging. We evaluate our protocol for different purging schemes as described before. Here, we describe actual purging process for different schemes. Apart from these schemes, remaining packets of a generation are purged from all the buffers after timeout period 'TTL'. This timeout mechanism ensures that, if a generation is not delivered to all the destinations or delivery information from all destinations are not received by a node even after TTL delay, it will be purged. So, value of TTL is to be kept sufficiently large such that generations are received by most of the destinations most of the time. The protocol maintains average number of destinations who have received a generation till TTL. We call it $avgDel$. This value is updated whenever a generation is expired or is delivered to all destinations.

1) Probabilistic Purging (PBP): In this scheme, when two nodes come into contact, they exchange accumulated lists of destinations which have received the entire generation for each generation. For each destination of a generation listed in the received list which is not in the node's list, entire generation is purged with probability $1/avgDel$ where $avgDel \leq TotalDest$. The idea behind using $avgDel$ instead of $TotalDest$ is, as only $avgDel$ number of destinations on an average will be able to receive a generation any way, there is no point in keeping buffers occupied once $avgDel$ number of destinations have received a generation.

2) Proportional Purging (PPP): Similar to PBP, in this scheme, when two nodes come into contact, they exchange accumulated lists of destinations which have received the entire generation for each generation. For each destination of a generation listed in the received list which is not in the node's list, the node reduces the $rank$ of the generation by following formula where number of destinations in the node's destination list is denoted by $TotalRecv$:

$$\begin{aligned} rank^{'} &= \frac{rank * (avgDel - TotalRecv)}{(avgDel - (TotalRecv - 1))} \\ rank &= \left\lfloor rank^{'} \right\rfloor \end{aligned} \tag{1}$$

Further, it increases the $rank$ by one with probability $rank^{'} - \left\lfloor rank^{'} \right\rfloor$.

3) Aggressive Purging (AP): In this scheme, when two nodes come into contact, they exchange accumulated information about $rank$ of generations at different destinations with each other. After receiving this list, the node compares the $rank$ (of a generation at a destination) received from the neighbour with the $rank$ for the same stored in it. If the received $rank$ is greater, then it updates the $rank$ as described in Algorithm 1 where $Ndrank_i$ denotes newly received value of $rank$ of a generation at a destination i and $Odrank_i$ denotes previously stored value of $rank$ of a generation at a destination i.

Algorithm 1: Purging algorithm

```
diff = 0
for each destination i do
    if Ndrank_i > Odrank_i then
        diff = diff + (Ndrank_i − Odrank_i);
        Odrank_i = Ndrank_i;
    end
end
factor = (diff * rank)/(G * avgDel);
rank = rank − ⌊factor⌋;
```

It further reduces the $rank$ by one with probability $factor - \lfloor factor \rfloor$.

C. Intelligent Beaconing

After purging, the node does intelligent beaconing. i.e., it computes $rank$ of all generations stored at the node and sends the coefficient matrix of the generations to the neighbour node provided the value of $token$ for a generation is greater than zero or the neighbour node is the destination node of the generation.

Along with the coefficient matrices, the node also sends values of $rank$ and $token$ of each generation to the neighbour node. We will denote these values as $frank$ and $ftoken$ at the neighbour node. This information is used to update the value of $token$ for each generation as described later.

The neighbour node, for each generation, appends the received coefficient matrix to the matrix it has for that generation such that number of rows in the augmented matrix is at most equal to G. Then, it uses Gauss elimination method to determine $rank$ of the augmented matrix. If it is greater than $rank$ of the original matrix, the difference denotes number of innovative packets the initiator node has for a particular generation at the neighbour node. To find total number of such innovative packets for the generation, this process is repeated till the $rank$ of the augmented matrix is less than G and there are remaining rows in the received coefficient matrix, which were not considered previously as the augmented matrix was full. The number of innovative packets thus found for each generation is sent to the initiator node. This number denotes the number of encoded packets the initiator node should send for a generation. Further, the node can at most send number of packets of a generation equal to the value of $token$ of that generation at that node. So, the node will try to send number of encoded packets equal to the minimum of the number of packets requested by the neighbour node and the value of $token$. Each of these encoded packets will increase the $rank$ of the corresponding generation at the neighbour node with high probability.

D. Updation of token

The value of $token$ for each generation at a node denotes the number of encoded packets of the generation to be forwarded by that node and it is updated before doing intelligent beaconing using following formula:

$$token = \frac{(token + ftoken) * rank}{(rank + frank)} \tag{2}$$

The idea is that after exchanging and updating $token$ at two nodes, number of encoded packets to be generated per independent encoded packet should be same at two nodes for all generations.

E. Packet Mixing and Forwarding

After intelligent beaconing, the initiator node sends encoded data packets to the neighbour till it is within communication range. The generation of which next encoded packet is to be forwarded, along with its coefficients, is decided based on following forwarding policy:

- All the generations are grouped into three classes. First class consists of the generations for which the node is the source. Second class is of the generations for which the neighbour is the destination node. Finally, the remaining generations are in the third class. These classes are considered for forwarding in the order in which they are mentioned.
- For each of the above three classes, an encoded packet is generated of the generation which has the maximum number of encoded packets to be sent. The encoded packet is the 'mixture' of all the independent packets stored at the node for the generation. After sending the packet, the node increments by one the value of $frank$ and decrements by one the value of $token$ of the corresponding generation.

F. Packet Reception

When the neighbour node receives the packet and the buffer is full, it makes room in the buffer as per the buffer management policy mentioned below and it stores the coefficients of the packet in the corresponding coefficient matrix and decrements by one the value of $ftoken$ received from the initiator node of the corresponding generation.

When the neighbour node receives the packet and the buffer is full then room for the packet is created by reducing $rank$ of one of the generations by one based on following buffer management policy:

- Choose the generation having maximum $rank$. If there are more than one such generations then
- Choose the one which is the oldest. If there are more than one such generations then
- Choose the one for which $token$ is lowest. If there are more than one such generations then
- Choose one of them randomly

If the $rank$ of chosen generation is greater than 1, two encoded packets of the generation are combined into one. Otherwise, the $rank$ is made 0, effectively dropping the packet. Please note that, this will always be the case in non-network coding based scheme.

G. Packet Decoding

When the $rank$ of a generation at the destination node is equal to G, it decodes G number of original packets from the generation.

V. Simulation Results

A. Simulation Environment

We have simulated the proposed protocol in NS2 simulator. The network contains 20 wireless nodes which move according to Random Way Point mobility model (RWP). The minimum speed of a node is 1 m/s and maximum speed is 20 m/s with average speed of around 6 m/s and pause time is zero. The communication range of a node is 100 m and bandwidth of the channel is 1 Mbps. Unless otherwise mentioned, number

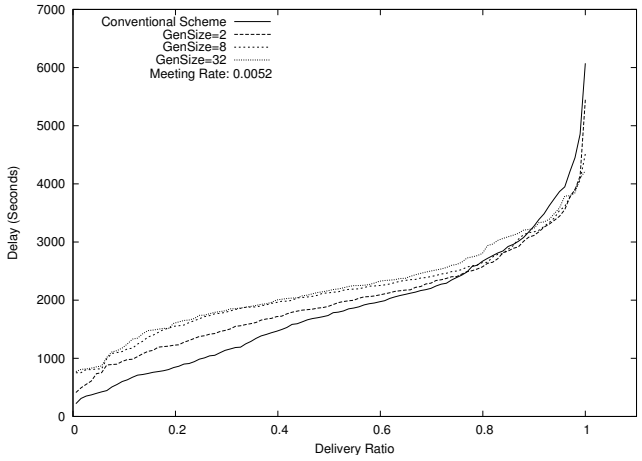

Fig. 3. Delay v/s Delivery ratio

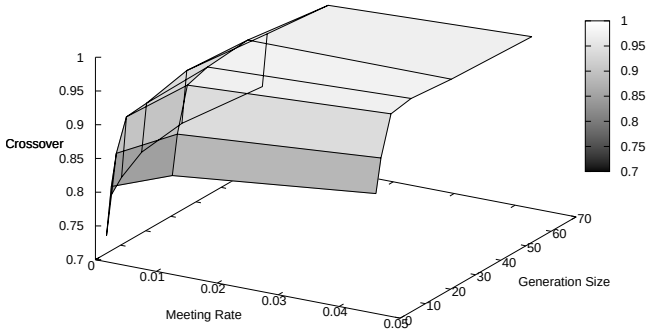

Fig. 4. Crossover ratio v/s Meeting rate and Generation size

of copies (L) of a packet in the network is 7 because as shown in Fig. 7, increasing L further does not decrease delay significantly. We define Meeting rate (γ) as average number of times a node meets with any other node in the network per second. It is changed by varying field area of the network. There are four sources in the network and for each source, there are four destinations. For network coding, randomly chosen coefficients and addition and multiplication operations for encoding and decoding are over the field $\mathbf{F}_{2^8}$. We compare our protocol with modified BBR [14] in which instead of unrestrictive flooding, we restrict number of copies per packet (L) and we spread them using binary spraying mechanism [11]. We will call it 'conventional scheme'. We have simulated the modified BBR by setting $G = 1$ in our protocol, which effectively disables the network coding, and by disabling purging mechanism. We compare our protocol with BBR as it is the only Multi-copy protocol to multicast in DTN.

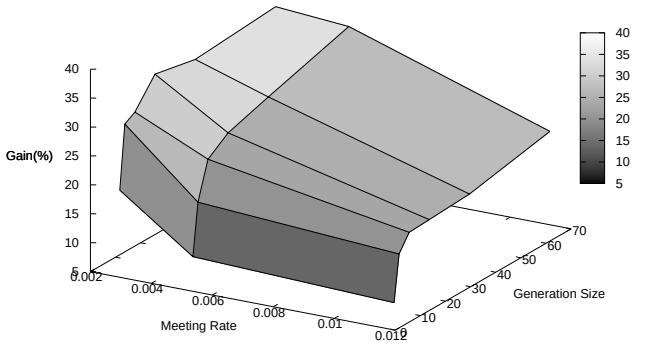

Fig. 5. Percentage gain in delay to deliver all the packets with respect to conventional scheme v/s Meeting rate and Generation size

B. Infinite Buffer Case

In this section, we present the performance of the protocol when buffers are infinite. i.e., each node has enough buffer to store all the generations from all sources. Once average speed of the nodes in the network achieves steady state, all the source nodes generate given number of data packets which are grouped into generations. i.e., data packets are generated in a pulse and there is no new traffic generated after that. Number of generations depend on size of generation. For the results shown, number of data packets per source are 704 and each packet is of 4500 bytes. We run the simulation for sufficient time period such that all the generations are received by all intended destination nodes. For infinite buffer case, anti-packets are generated only when entire generation is delivered. A node purges entire generation when the node has accumulated all the destinations in its destination list for that generation.

We observe that for lower value of α, conventional scheme outperforms our protocol. We define 'crossover' point as delivery ratio (α) after which our protocol has lower delivery delay (σ) than the conventional scheme. As shown in Fig. 3, lower G outperforms conventional scheme earlier (at lower α) than higher G. But then, higher G outperforms lower G and delay to deliver all the packets is lesser as G increases. Please note that the result is for $\gamma = 0.0052$. The reason behind this behaviour is as follows: When number of encoded packets received at destinations is low, for smaller G, some of the generations can be decoded early and the α is more than the α of higher G even though number of independent packets received for higher G are more. But as the number of independent packets increases, larger generations can also be decoded and as the number of decoded packets from a generation are more for higher G, α increases faster and it outperforms the α for lower G. Further, σ increases very fast for higher α starting from say 0.8. It is because as the α increases, the probability of finding required packet of a generation decreases very fast. The rise in σ is not so steep in our protocol because every encoded packet of a generation carry information of the required packet and this number

increases with increase in G.

As shown in Fig. 4, as the γ decreases, our protocol for different G has lower crossover point. It is because for lower γ, number of packets to be forwarded by a node when it meets another node is low and so the probability of finding unique packet is also low.

Fig. 5 shows percentage gain in delay to deliver all the packets with respect to conventional scheme for different G and γ. As evident from the figure, with decrease in γ and increase in G, gain increases significantly. Further, after a particular G, say 16 in this case, with increase in G, gain is not significant.

Fig. 7 shows Delay to deliver all the packets v/s Number of copies per packet in the network (L) for conventional scheme and for network coding with $G = 8$. Please note that the result is for $\gamma = 0.0076$. This result is similar to the one given by Spyropoulos et al. in [11] and it says that initially with increase in L in the network, delivery delay (σ) decreases quite significantly but as the L increases further, improvement is not that significant. The reason for this behaviour is as follows: As we increase the L in the network, i.e., more number of nodes in the network have a copy of a packet, chances of one of them meeting the destination increases. But with increase in L, time between encounters of a destination node and the nodes having copy of the packet decreases and as a result improvement in σ decreases.

Further, it is also evident from Fig. 7 that to achieve delay to deliver all the packets same as in conventional scheme, the protocol requires less number of copies per packet.

C. Estimation of Generation Size (G)

As discussed earlier, conventional scheme has lower delivery delay (σ) than our protocol upto some α, say 0.8 (Fig. 3). If the application requires α greater than this, rate of increase in σ with increase in α is very high for conventional scheme and this rate (denoted by Delay-delivery ratio coefficient $m = \frac{d\sigma}{d\alpha}$) decreases with increase in G. So, for applications requiring higher α with given delay bound, our protocol with different values of G outperforms conventional scheme.

Fig. 6 shows Delay-delivery ratio coefficient v/s Generation size for different meeting rates. From this empirical result, we have derived following relation between delay-delivery ratio coefficient (m), generation size (G) and meeting rate (γ).

$$\begin{aligned} m &\propto -\frac{\gamma}{\sqrt{G}} \\ G &\propto \left(\frac{\gamma\alpha}{\sigma}\right)^2 \end{aligned} \tag{3}$$

From the relation, for the given meeting rate, we can estimate optimal generation size for desired delivery ratio and delay bound.

D. Finite Buffer Case

When buffers of the nodes are limited, forwarding nodes will not able to store all the innovative packets it receives. So, limiting buffer usage is very important. Here, we compare

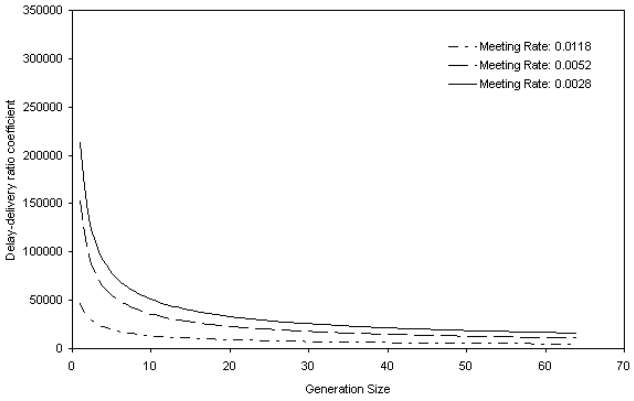

Fig. 6. Delay-delivery ratio coefficient v/s Generation size

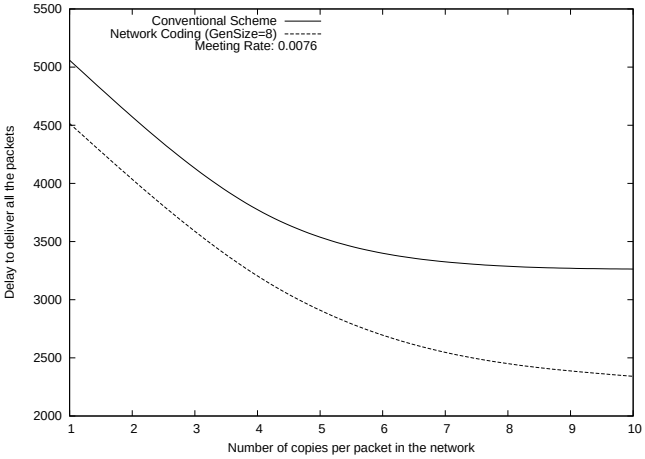

Fig. 7. Delay to deliver all the packets v/s Number of copies per packet in the network

delivery ratio (α) of conventional scheme which does not do purging and our protocol with different purging schemes. Please note that whenever an innovative packet is received, a node will drop a packet from its buffer according to the buffer management policy explained earlier.

We ensure that source nodes have effectively infinite buffers by keeping buffer size very large while forwarding nodes have finite buffers. Once average speed of the nodes in the network achieves steady state, source nodes start generating Constant Bit Rate (CBR) traffic with the rate of $1/12$ packets per second where size of one packet is 4096 bytes.

As shown in Fig. 8, once the network reaches steady state, α of our protocol compared to conventional scheme is higher with any of the three purging schemes. Among different purging schemes, PPP gives highest α. Please note that the result is for $\gamma = 0.012$. The reason for this behaviour is as follows: AP purges too many independent packets early and as a result, destination nodes are not able to receive enough number of independent packets even though it reduces number of dropped packets due to buffer overflow considerably. While PBP and PPP purges similar amount of independent packets (only when entire generation at a destination is received), PBP

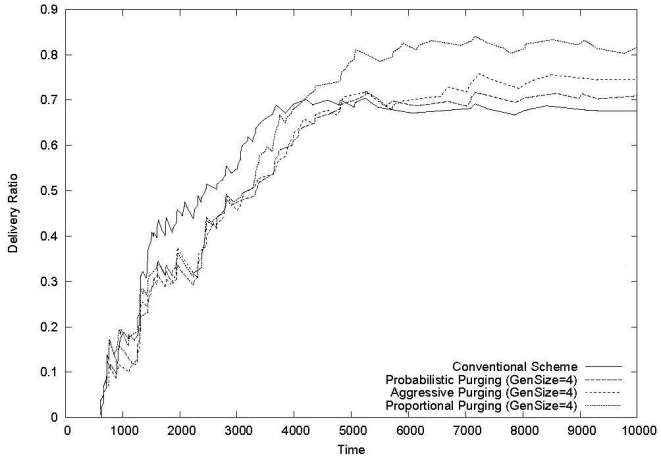

Fig. 8. Delivery ratio v/s Time

purges entire generation from a node (with some probability). So that node is not able to contribute to eventual delivery after that. It would work better when nodes move in form of clusters and so when a destination node in a cluster receives a generation, forwarding nodes in that cluster can purge that generation from their buffer(with some probability) because any way, they will not able to deliver packets to destination nodes in other clusters. Based on this result, we choose to use PPP in our protocol.

VI. Conclusion

Delay Tolerant Networks require different routing strategy than conventional networks because of frequent and long partitions in the network. To improve chances of delivery, Multi-copy schemes are used. However, communication overhead and buffer occupancy increases as we increase number of copies per packet. To reduce this overhead without impacting performance, we use network coding based Multi-copy scheme. We have also introduced a novel packet purging scheme which takes advantage of features of network coding to increase buffer efficiency.

The major contributions of our work are as follows:

- Protocol design to multicast in DTN with network coding and an improved purging scheme
- Buffer management policy for finite buffer nodes

Summary of major findings of our work are as under:

- Simulation results prove that the protocol reduces delay to deliver all the packets by 15% to 30%.
- For the infinite buffer case, the protocol outperforms conventional scheme for delivery ratio greater than 80%. The protocol with lower generation size outperforms conventional scheme at lower delivery ratio but at higher delivery ratio, higher generation size outperforms lower generation size.
- Improvement of the protocol over conventional scheme is more pronounced as the network becomes sparser.
- Initially with increase in generation size, delivery delay decreases quite significantly but as the generation size increases further, improvement is not that significant.
- To achieve delay to deliver all the packets same as in conventional scheme, the protocol requires less number of copies per packet.
- For the finite buffer case, delivery ratio can be improved by using effective purging scheme with network coding.

References

[1] P. Yang and M. C. Chuah, "Context-aware multicast routing scheme for disruption tolerant networks," in *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pp. 66 - 73, 2006.

[2] C.-C. Chiang, M. Gerla, and L. Zhang, "Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks," in *Cluster Computing*, vol. 1(2), pp. 107 - 120, 1998.

[3] S. Lee, M. Gerla, and C. Chiang, "On-demand multicast routing protocol (ODMRP) for ad hoc networks," in *Internet draft*, June 1999.

[4] J. Moy, "Multicast extensions to OSPF," in *RFC 1584*, March 1994.

[5] E. Royer and C. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *ACM Mobicom*, August 1999.

[6] Z. Guo, P. Xie, J. Cui, and B. Wang, "On applying network coding to underwater sensor networks," in *WUWNet*, September 2006.

[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," in *IEEE Transactions on Information Theory*, vol. 46, pp. 1204-1216, July 2000.

[8] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," in *IEEE Transactions on Information Theory*, vol. 49, pp. 371-381, February 2003.

[9] T. Ho, R. Koetter, M. Mèdard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory*, 2003.

[10] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," in *Technical Report CS-200006, Duke University*, April 2000.

[11] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252 - 259, 2005.

[12] T. Small and Z. J. Haas, "Resource and performance tradeoffs in delay-tolerant wireless networks," in *Proceeding of the ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 260 - 267, 2005.

[13] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "On the benefits of random linear coding for unicast applications in disruption tolerant networks," in *Fourth International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, April 2006.

[14] M. A. Wenrui Zhao and E. Zegura, "Multicasting in delay tolerant networks: Semantic models and routing algorithms," in *Proceedings of the ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[15] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "On the benefits of network coding for wireless applications," in *Second Workshop on Network Coding, Theory, and Applications*, April 2006.

[16] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard, "Codecast: a network-coding-based ad hoc multicast protocol," in *IEEE Wireless Communications*, vol. 13, pp. 76-81, October 2006.

[17] Y. Xi and E. M. Yeh, "Distributed algorithms for minimum cost multicast with network coding in wireless networks," in *4^{th} International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, April 2006.

[18] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *First International Conference on Wireless Internet*, July 2007.

[19] D. S. Lun, N. Ratnakar, M. Mèdard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," in *IEEE Transactions on Information Theory*, vol. 52, pp. 2608-2623, June 2006.

[20] Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," in *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 107 - 120, January 2004.