# Integrating Random Network Coding with On-Demand Multicast Routing Protocol

Joon-Sang Park and Seung Jun Baek

*We propose integrating random network coding with the Enhanced On-Demand Multicast Routing Protocol (E-ODMRP). With the Network Coded E-ODMRP (NCE-ODMRP), we present a framework that enables a seamless integration of random linear network coding with conventional ad hoc multicast protocols for enhanced reliability. Simulation results show that the NCE-ODMRP achieves a nearly perfect packet delivery ratio while keeping the route maintenance overhead low to a degree similar to that of the E-ODMRP.*

*Keywords: Ad hoc networks, network coding, multicast.*

## I. Introduction

The Enhanced On-Demand Multicast Routing Protocol (E-ODMRP) [1] is an ad hoc multicast routing protocol featuring an adaptive route refreshing mechanism. Its predecessor, the ODMRP [2], is renowned for robustness to mobility. The ODMRP's robustness comes from the periodic route refreshing. The ODMRP rebuilds the data forwarding mesh on a short fixed interval, which is the main cause of its high overhead. The E-ODMRP was proposed to reduce the overhead of the ODMRP through an adaptive refreshing mechanism: the forwarding mesh refresh interval is adjusted to the network condition. Thus, the major advantage of the E-ODMRP is the reduction of overhead by up to 90% compared to that of the ODMRP. However, such reduction compromises reliability. In this letter, we propose a new protocol, the Network Coded E-ODMRP (NCE-ODMRP), which integrates random network coding with the E-ODMRP for enhanced reliability.

Network coding [3] has been shown to enhance various performance metrics of wireless networks, such as throughput and reliability. In CodeCast [4], it has been shown that using network coding ad hoc multicast could achieve near-100% delivery ratio. However, the drawback of CodeCast is that it cannot be integrated with existing ad hoc multicast protocols.

In this letter, we introduce a framework that enables seamless integration of random network coding [5] with existing ad hoc multicast routing protocols for enhanced reliability. The key techniques we use include variable-size generations and low-delay encoding schemes adopted in online network coding [6] and Combo/Pipeline coding [7]. We show via simulation that the NCE-ODMRP enhances the packet delivery ratio yet keeps the overhead the same compared to that of the E-ODMRP. Among a number of ad hoc multicast protocols proposed in the literature, the E-ODMRP is chosen as the base protocol since it is one of the best performing protocols. Recently, [8] showed that XOR coding can help reduce control overhead in the ODMRP. In [8], it is suggested that two of the control packets are XORed before being transmitted, whenever possible; however, such a simple XOR scheme cannot improve the reliability of any existing protocol. In our scheme, aside from some control packets, every transmitted packet is encoded using random network coding to simultaneously enhance reliability and keep control overhead low.

## II. NCE-ODMRP Protocol

### 1. Overview of E-ODMRP

When a source node has multicast data to send, the node starts with flooding a join query (JQ). A JQ packet is a data

Manuscript received Feb. 6, 2012; revised May 31, 2012; accepted June 15, 2012.
This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (2010-0005334).
Joon-Sang Park (phone: +82 2 320 3033, jsp@hongik.ac.kr) is with the Department of Computer Engineering, Hongik University, Seoul, Rep. of Korea.
Seung Jun Baek (sjbaek@korea.ac.kr) is with the Division of Computer and Communications Engineering, Korea University, Seoul, Rep. of Korea.
http://dx.doi.org/10.4218/etrij.12.0212.0057

  

packet with a special flag. Upon receipt of the first non-duplicate JQ packet, every node sets pointers to its upstream node and rebroadcasts it. Once the JQ reaches a receiver, the receiver sends a join reply back to the source. The join reply is relayed by the intermediate nodes to the source following the pointers set when the JQ was propagated. The nodes that have relayed the join reply become forwarders forming a forwarding mesh. The source refreshes the forwarding mesh on variable-interval schedules.

An intermediate node or a receiver can be disconnected from the mesh due to mobility. The multicast source estimates the inter-packet arrival times and informs receivers by including the estimate in JQ packets. If a node in the mesh does not receive any data for a certain multiple of the reported packet interarrival times, the node considers itself to be detached from the mesh and performs a recovery procedure. Upon detection of a broken route, a node broadcasts a receiver join (RJ) packet to its neighbor. When a listener node, defined to be a neighbor of any forwarder or receiver, receives the RJ packet, it becomes a temporary forwarder (TF) and immediately starts forwarding data packets. There may exist multiple TFs; if so, the receiver chooses one of them using an implicit ACK as a regular forwarder, which becomes a member of the forwarding group associated with the receiver. This local search to find TFs may be repeated three times. If a node fails to graft to the mesh through the local searches, the node floods a refresh request. The initiator of the refresh request estimates the route lifetime, which is the difference in time between the route establishment and the detection of route breakage, and records it in the route refresh packet to be passed to the source via network-wide flooding. The source adjusts its JQ flooding rate as the reciprocal of the route lifetime received in a refresh request packet and floods a JQ.

## 2. NCE-ODMRP

In this subsection, we describe how the NCE-ODMRP integrates random network coding with the E-ODMRP. Suppose a source node generates a stream of equally-sized packets $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$,…, where the subscripts denote unique and consecutive sequence numbers (SNs) in $\mathbb{Z}_+$. The source packets are called frames, and they can be considered as vectors over a finite field $\mathbb{F}$. The stream of frames is logically partitioned into a set of generations, each of which is a set of frames with adjacent SNs. Each generation is associated with a generation ID (GID), which is a unique ID defined as the smallest SN of the frames belonging to the generation. We will use GIDs to denote generations. For a given GID $g$, a frame $\mathbf{p}_k$ is said to be in the generation $g$, if $g \le k < g+u_g$, where $u_g$ denotes the size of the generation $g$. Note that the generations may vary in size; that is, $u_g$ may differ depending on $g$. If we denote the generation that follows generation $g$ by $\overline{g}$, then $\overline{g} = g+u_g$. Due to varying $u_g$, GIDs are not consecutive SNs in general.

**Encoding.** When the frame $\mathbf{p}_s$ is generated and passed to the routing layer from the application, the routing layer generates a coded packet $\mathbf{c}_{(g,s)}$ using the previously stored frames that share the same GID $g$ with $\mathbf{p}_s$ and broadcasts $\mathbf{c}_{(g,s)}$ to the neighbors. The coded packet $\mathbf{c}_{(g,s)}$ is a linear combination of the frames in the generation $g$, $\mathbf{c}_{(g,s)} = \sum_{k=1}^{s-g+1} e_k \mathbf{p}_{k+g-1}$, where $e_k$ is drawn according to the uniform distribution over $\mathbb{F}$ [5]. In the header of a coded packet $\mathbf{c}_{(g,s)}$, the encoding vector $\mathbf{e}=[e_1 \ldots e_{s-g+1}]$ is stored along with the GID $g$ and the SN $s$. Namely, $s$ is the largest among the SNs of the frames that are used to generate $\mathbf{c}_{(g,s)}$. Note that the routing layer does not wait for a generation to be amassed to generate coded packets. Instead, a coded packet is constructed and transmitted upon the arrival of every new frame. The construction of a coded packet is done by combining frames belonging to the same generation collected up to the point of construction. For example, when $\mathbf{p}_1$ is passed to the routing layer, a coded packet $\mathbf{c}_{(1,1)}=e_1\mathbf{p}_1$ is transmitted. Subsequently, as soon as $\mathbf{p}_2$ is available, $\mathbf{c}_{(1,2)}=e_1\mathbf{p}_1+e_2\mathbf{p}_2$ is transmitted, and, similarly, a coded packet $\mathbf{c}_{(1,3)} = \sum_{k=1}^{3} e_k \mathbf{p}_k$ is sent out when $\mathbf{p}_3$ becomes available. Each encoding vector can be viewed as a compressed form in which zero elements are eliminated. For instance, the encoding vector of a coded packet $\mathbf{c}_{(1,2)}$, $[e_1e_2]$, must be expanded to $[e_1e_20 \ldots 0]$ when decoding. The number of zero elements depends on how many packets are decoded simultaneously, which is to be explained later.

The main difference between our encoding scheme and online network coding [6] is the necessity of a feedback mechanism: online network coding assumes the existence of a reliable feedback mechanism, whereas we do not rely on any feedback. In our case, the coding window, that is, the range of frames used to generate coded packets, moves on the arrival of every new frame instead of when receiving feedback from receivers as in online network coding. Another difference is the use of generations. The idea of constructing a coded packet using packets collected up to the point of construction was explored in [7].

The forwarding operation performed by a node receiving a coded packet $\mathbf{c}_{(g,i)}$ is illustrated in Fig. 1. Forwarding nodes must generate and broadcast a locally reencoded packet $\mathbf{c}'_{(g,i)}$ to their neighbors when they receive a coded packet tagged with a new tuple $(g, i)$. The locally reencoded packet $\mathbf{c}'_{(g,i)}$ is a random linear combination of the packets tagged with the same GID $g$ available in the local memory. Let $\mathbf{c}_k$ be the coded packets with the GID $g$ in the local memory and $\mathbf{e}_k$ be the encoding vector

Upon receiving coded packet $\mathbf{c}_{(g,i)}$
  **Store** $\mathbf{c}_{(g,i)}$ in local memory.
  If $(g, i)$ is not in reception list
    **Store** $(g, i)$ in reception list.
    If ($\mathbf{c}_{(g,i)}$ is JQ) or (I am a forwarder or TF)
      **Generate** and **broadcast** a locally reencoded frame $\mathbf{c}'_{(g,i)}$.

Fig. 1. Forwarding operation.

prefixed to $\mathbf{c}_k$. We construct the reencoded packet $\mathbf{c}'_{(g,i)} = \sum_{k=1}^{n} e'_k \mathbf{c}_k$ and calculate its encoding vector $\mathbf{e}' = \sum_{k=1}^{n} e'_k \mathbf{e}_k$, where $n$ is the number of coded packets with the GID $g$ and $\{e'_k\}$ are drawn uniformly and randomly from $\mathbb{F}$. To save the usage of local memory, it can be flushed when a node detects the start of a new generation.

**Generations.** Once the concept of generation is used, generation management becomes an important issue. In our protocol, we use variable-size generations to accommodate how frequently the route refresh occurs. The partitioning of generations is carried out on the fly. Specifically, the generation size is determined as the number of frames generated by the multicast source between the two successive occurrences of JQ flooding. A new generation begins with the first frame transmitted following a JQ packet that is flooded for the route refreshment. The generation ends with the next JQ packet. For example, if $\mathbf{p}_i$ in the generation $g$ is generated by the application and the corresponding coded packet $\mathbf{c}_{(g,i)}$ is flooded to refresh routes, then $\mathbf{p}_{i+1}$ starts a new generation and the generation $g$ ends with $i$. Again, every time a new frame becomes available, it is stored in the buffer. Then, the routing layer combines all the stored frames in the buffer to generate and transmit a coded packet. To prevent combining packets from different generations, the buffer is flushed after every JQ.

Every packet carries a placeholder for the largest encoding vector (120 bytes), and the generation size is not allowed to exceed the largest encoding vector size: once the generation size reaches this limit, a route refresh is enforced.

**Decoding.** Upon receipt of a coded packet $\mathbf{c}_{(g,s)}$, nodes store the packet in their local memory and attempt to decode it whenever possible instead of waiting for the entire generation to be accumulated. If there is no packet loss, $\mathbf{p}_s$ can easily be recovered as soon as $\mathbf{c}_{(g,s)}$ is received; for example, $\mathbf{p}_{g+1}$ can easily be decoded from $\mathbf{c}_{(g,g)}$ and $\mathbf{c}_{(g,g+1)}$. Despite packet loss, the frames can still be recovered if a sufficient number of coded packets with duplicate SNs in the same generation are collected. Note that due to the loss recovery procedure and/or flooding operations done for route refreshing, there are chances of receiving coded packets tagged with the same tuple $(g, i)$. Specifically, when $\mathbf{c}_{(g,s)}$ is received, the frame $\mathbf{p}_s$ can be recovered if the following conditions hold: $N_{g+k-1} \geq s-g-k+2$ for $k=1,\ldots, s-g+1$, where $N_j$ is the number of coded packets stored in the local memory, which are tagged with the GID $g$ and the SN no less than $j$ and are associated with encoding vectors that are linearly independent of each other. The decoding process is as follows. For a given SN $i$, let $\mathbf{c}_k$ be a coded packet with the GID $g$ and with an SN less than or equal to $i$ in the local memory, $\mathbf{e}_k$ be the encoding vector prefixed to $\mathbf{c}_k$, and $\mathbf{p}_{g+k-1}$ be the frame to be recovered, where $k=1,\ldots, i-g+1$. Since $\mathbf{e}_k$'s may have different lengths, we construct $\bar{\mathbf{e}}_k$'s with a constant length of $i-g+1$ by appending zeros to $\mathbf{e}_k$'s; for example, if $\mathbf{e}_k=[e_1 e_2]$ and $i-g+1=5$, then $\bar{\mathbf{e}}_k = [\mathbf{e}_1 \mathbf{e}_2 000]$. Now, let $\mathbf{E}^T = [\bar{\mathbf{e}}_1^T \ldots \bar{\mathbf{e}}_{i-g+1}^T]$, $\mathbf{C}^T = [\mathbf{c}_1^T \ldots \mathbf{c}_{i-g+1}^T]$, and $\mathbf{P}^T = [\mathbf{p}_g^T \ldots \mathbf{p}_i^T]$. We obtain the original frames from $\mathbf{P}=\mathbf{E}^{-1}\mathbf{C}$, assuming $E$ is invertible.

**Loss Recovery.** When a node detects route breakage in E-ODMRP, the node broadcasts an RJ with the hope of finding TFs that can connect the node to the mesh. We augment an RJ with a GID $g$ and the SN $s$ that is the largest of the packet SNs received up to that point by the transmitter of the RJ. Upon receipt of the RJ, each listner node (after changing its status to TF) transmits its own version of coded packet $\mathbf{c}_{(g,s)}$ so that the transmitter of the RJ can collect a sufficient number of coded packets, which facilitates recovering any lost frames. This local recovery scheme may not recover all of the lost frames. However, occassionally, the multicast source floods the entire network with a JQ for the purpose of reconstructing the forwarding mesh. Each node receives an abundance of duplicate packets when flooding occurs, which serves as an opportunity to recover further lost packets. Since the flooded packet is the last packet in a generation, it is highly likely that nodes can collect a sufficient number of coded packets to decode the entire generation. Most importantly, duplicate JQ packets are deemed useless in the E-ODMRP and the ODMRP, whereas they serve as a useful means to the recovery of lost packets in the NCE-ODMRP.

## III. Performance Evaluation

In this section, we evaluate the performance of the NCE-ODMRP using ns-2 with the following parameters: 802.11 MAC; transmission range of 250 m; channel bandwidth of 2 Mbits/s; two-ray ground reflection model; constant bit rate traffic of four 512-byte packets/s; 100 nodes randomly placed on a 1200 m × 800 m field; single multicast group with single source and 20 receivers; random waypoint mobility model with 0 pause time, 1-m/s minimum speed, and varying maximum speed; 900 seconds of simulation time; and E-ODMRP and ODMRP defaults. Results are averaged over 20 runs with independent random seed numbers. Simulations are

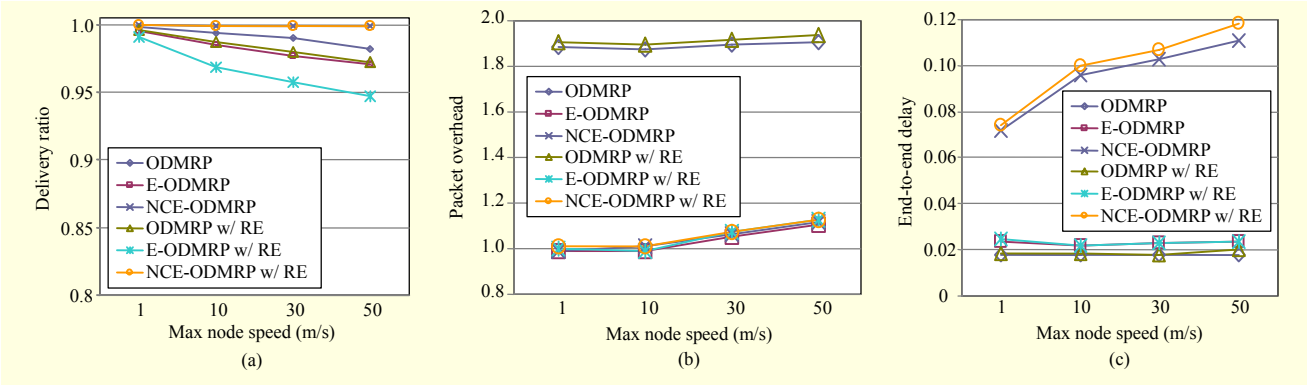


Fig. 2. Comparison of (a) delivery ratio, (b) overhead, and (c) end-to-end delay of three protocols.

performed with and without random errors (REs). To simulate REs, nodes are enforced to drop successfully received packets randomly and uniformly with a probability of 0.1. Besides REs, packet losses can occur in simulations due to packet collisions and node mobility.

In Fig. 2(a), the NCE-ODMRP shows nearly 100% data delivery regardless of node mobility and channel condition, that is, the existence of REs. The E-ODMRP shows a delivery ratio of 97% (without REs) and 95% (with REs) when the max node speed is 50 m/s. The delivery ratio is defined as the ratio of the total number of packets successfully received by the receivers over the total number of packets generated by the source. We observe that the NCE-ODMRP fails to achieve 100% delivery only in the rare case that the network stays partitioned for a long period.

As shown in Fig. 2(b), the overhead of the NCE-ODMRP is far less than that of the ODMRP and is very close to that of the E-ODMRP. To measure the overhead of a protocol, we use a metric called "packet overhead," defined as the total number of packets transmitted divided by the total number of frames delivered to any receiver. The packet overhead with REs is somewhat higher than the overhead without REs, but the difference is less than 1% for all three protocols. From the figure, we can see that the NCE-ODMRP achieves enhanced reliability without compromising the spectral/bandwidth efficiency compared to the E-ODMRP. It is worth noting that the NCE-ODMRP's overhead is slightly lower than that of the E-ODMRP in some cases.

An end-to-end delay is defined as the difference between the packet generation time at the source and the packet delivery time at the receiver. In Fig. 2(c) the NCE-ODMRP's end-to-end delay is slightly higher than that of the other protocols due to its packet recovery process. Note that these delays are dominated by a small number of lost frames, which are recoverable by the NCE-ODMRP but would be permanently lost under other protocols. As mentioned in the previous section, if a packet is missing, the corresponding frame is recovered later when a sufficient number of packets in the same generation are collected.

## IV. Conclusion

In this letter, we presented an On-Demand Multicast Routing Protocol with network coding for ad hoc networks called NCE-ODMRP. In our simulation study, NCE-ODMRP showed near perfect packet delivery ratio while maintaining less or similar overhead compared to that of existing multicast protocols.

## Acknowledgement

The authors are grateful to H. You for his help in simulations.

## References

[1] S. Oh, J.-S. Park, and M. Gerla, "E-ODMRP: Enhanced ODMRP with Motion Adaptive Refresh," *J. Parallel Distrib. Comput.*, vol. 68, no. 8, 2008.

[2] S.-J. Lee, W. Su, and M. Gerla. "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks," *Mobile Netw. Appl.*, vol. 7, no. 6, 2002.

[3] R. Ahlswede et al., "Network Information Flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, 2000, pp. 1204-1216.

[4] J.-S. Park et al., "Codecast: A Network Coding Based Ad Hoc Multicast Protocol," *IEEE Wireless. Commun.*, vol. 13, no. 5, 2006.

[5] T. Ho et al., "A Random Linear Network Coding Approach to Multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, 2006, pp. 4413-4430.

[6] J.-K. Sundararajan, D. Shah, and M. Medard, "Feedback-Based Online Network Coding," Submitted to *IEEE Trans. Inf. Theory*, 2009. http://arxiv.org/abs/0904.1730

[7] C.-C. Chen et al., "ComboCoding: Combined Intra-/Inter-flow Network Coding for TCP over Disruptive MANETs," *J. Adv. Res.*, vol. 2, no. 3, 2011, pp. 241-252.

[8] T. Yang et al., "Improved Network Coding Based on ODMRP Protocol in Ad Hoc Network," *Proc. ICACC*, 2011.