Rainbow: A Wireless Medium Access Control Using Network Coding for Multi-hop Content Distribution

Chen-Mou Cheng, H. T. Kung, Chit-Kwan Lin, Chia-Yung Su, Dario Vlah
Harvard University
Cambridge, Massachusetts

ABSTRACT

We consider the problem of multi-hop content distribution over a wireless ad-hoc network. Such mechanisms are relevant to a broad spectrum of applications, but are particularly important to data broadcast in wireless distributed computing where speedy I/O is critical to overall performance. In this paper, we present *Rainbow*, a content distribution protocol for multi-hop wireless ad-hoc networks. The protocol uses a *content-directed* medium access control (MAC), through which transmission priority is given to those nodes most capable of delivering useful content to their neighbors. We describe an efficient implementation of Rainbow based on network coding. Specifically, Rainbow uses a MAC priority scheme, where the priority of packet transmission from a node depends on the rank of the coefficient matrix associated with the coded content the node holds. We demonstrate that Rainbow achieves a 1.3- to 1.9-fold improvement in content distribution time over other flooding protocols, as measured on a testbed of 29 wireless nodes. We attribute this performance gain in part to Rainbow's ability to address a MAC-level bottleneck in multi-hop wireless networks, which we refer to as the "bridge lock-out problem".

1. INTRODUCTION

Content distribution—the problem of transmitting a large piece of data from a source node to all other nodes in a network as quickly as possible—has been widely studied in wire-line networks. Traditionally, it has been the purview of peer-to-peer (P2P) overlay schemes such as BitTorrent [1], Gnutella [2], Chord [3] and Pastry [4], where the wired network is assumed to be unicast, to have consistently low packet loss, and to have a topology experiencing a low churn rate.

In this work, we consider the content distribution problem in a different domain: wireless multi-hop ad-hoc networks. Potential applications of a protocol geared towards this scenario span a wide spectrum. In the military realm, an Unmanned Aerial Vehicle (UAV) might be called upon to distribute up-to-date, local surveillance images to multiple ground units. Another application would be in wireless distributed computing, related to our recent work in distributed speaker identification [5], where a wireless content distribution method can be used to broadcast input data for a distributed or parallel computation.

The contributions of this paper are as follows: (1) We introduce *Rainbow*, a protocol specifically designed for content distribution over multi-hop wireless ad-hoc networks. Rainbow uses a *content-directed* MAC and network coding [6] to improve broadcasting efficiency. To our knowledge, this is the first real-world implementation of content distribution over wireless multi-hop ad-hoc networks using network coding. (2) We compare Rainbow to two other flooding protocols and find that Rainbow gives a significant performance advantage over *content-blind* flooding and *uncoded* flooding in general topologies. Rainbow achieves this performance gain in spite of the computation and communication overhead incurred by network coding. (3) We empirically demonstrate the extent of the *bridge lock-out problem*, a type of wireless hidden-terminal problem that is particularly marked in multi-hop content distribution due to the heavy use of broadcasting. We also discuss how Rainbow addresses this problem.

2. CHALLENGES AND APPROACH

Properties specific to wireless multi-hop ad-hoc networks introduce additional complexity to the content distribution problem. The wireless medium is shared, rendering wire-line content distribution strategies that rely on multiple unicast flows inefficient. For example, we have confirmed that BitTorrent, which uses multiple, simultaneous TCP connections to peers, performs poorly on our wireless testbed. Similarly, the shared wireless medium also causes conventional flooding techniques (e.g., in [7]) to suffer from the "broadcast storm problem" [8], where relaying nodes experience heavy contention for the channel since their rebroadcast events are highly correlated in time. More

978-1-4244-2677-5/08/$25.00

advanced techniques that constrain flooding to a broadcast/multicast tree [9] require the building of a shortest path tree or a Steiner tree. However, physical phenomena such as fading, interference and capture effects [10] translate into transient changes in link quality that are difficult to predict, implying that the overhead of constantly rebuilding or re-ordering such trees can be prohibitively expensive.

Of particular note is the wireless-specific, hidden terminal problem [11]. In the classic case, hidden terminals cause collisions but, under the 802.11 carrier-sensing MAC, hidden terminals may even prevent certain nodes from *transmitting* for prolonged periods of time. In scenarios where there are multiple clusters of nodes and content must pass through bottleneck, "bridge" nodes, this can cause severe performance degradations. We term this phenomenon the *bridge lock-out problem*, and illustrate it using the simple scenario in Figure 1.

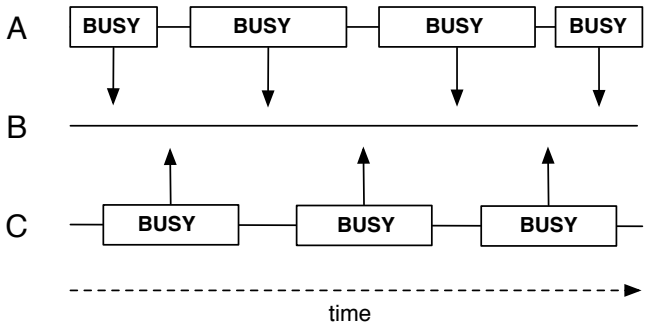

Figure 1. Illustration of the bridge lock-out problem in a three-cluster scenario, where content from a source node in cluster A is propagated to all other nodes. We assume that nodes are geographically located such that clusters A and C cannot sense each other's transmissions, but both have good links with B. Since the 802.11 contention periods of clusters A and C do not coincide, B will generally not sense an idle medium and thus will be "locked out" of sending.

We first observed the bridge lock-out problem on a real-world wireless testbed deployed in an office building and organized into a three-cluster topology. Simulating the phenomenon requires a detailed MAC model; we were able to reproduce it using the OPNET [12] discrete event simulator in a scenario where clusters A and C had 5 nodes each, and B contained 1 node.[1]

Despite the challenges outlined above, wireless offers the opportunity to exploit the medium's broadcast properties. Due to the nature of wireless channels, a single transmission can be heard by multiple receivers—a property we term *broadcast advantage*. In comparison to multiple unicasts, broadcast advantage has the potential to significantly reduce the number of transmissions required for distributing a piece of content to multiple receivers, even under lossy conditions.

Our general approach to wireless content distribution is to use broadcasts to propagate content over multiple hops, as would be done in most flooding techniques. However, the shared nature of the medium raises the question of how to allocate channel share to nodes so that they can relay efficiently. To address this issue, we use a notion called *innovative content* [13]. A piece of content is deemed to be "innovative" to a node if it represents a new piece of content which cannot be derived from existing content already held at the node. When network coding is used (see Section 3.2), innovative content will increase the rank of the coded content the node holds. Clearly, transmission priority should be given to those nodes which are most capable of delivering innovative content to their neighbors. To determine which are "most capable", a node's content and link quality, relative to each neighbor, should be taken into account. We call a MAC that follows this policy a *content-directed MAC*. In contrast, we term protocols that do not use such policies as "content-blind".

Even with a content-directed MAC, efficient implementation of a wireless content distribution protocol can still be difficult, as all nodes must receive the content in its entirety. Flooding protocols are faced with the "coupon collector's problem" [14]. That is, if any node is delayed in receiving its last piece of content, the entire content distribution is delayed. More sophisticated relay schemes such as Selective Forwarding [15] are susceptible to high inefficiency: if only one node is missing a particular piece of the content, then its broadcast retransmission is useless to all the other nodes that have already received it.

Network coding has been suggested as a way to overcome these limitations (e.g., [15], [16], [17], [13], [18]), as it has been shown to be an efficient reliable wireless multicast method which achieves a logarithmic reliability gain over ARQ mechanisms [19]. For this reason, and because our target environment is likely to have lossy wireless links, we employ network coding as a protocol building block.

A further benefit of network coding, particularly useful for the implementation of our content-directed MAC, is the convenience of using the rank information inherent to random linear coding [20] as a compact indicator of how much innovative content a node has, relative to its neighbors. Thus, rank serves as the lynchpin that ties together network coding (as an efficient and reliable wireless multicast method) with a content-directed MAC.

Our protocol, called *Rainbow*, is designed to address the kind of challenges and opportunities described above.

1. With the 11 Mbps 802.11b modulation, and infinite offered load on all nodes, nodes in clusters A and C achieved a 1.8 Mbit/s average transmission rate, while node B achieved merely 7.9 Kbit/s.

Specifically, Rainbow is a content-directed MAC protocol that makes novel use of network coding in improving efficiency and reliability in the MAC implementation, while minimizing transmission overhead. We have implemented Rainbow and validated the implementation on a testbed of 29 XO (Beta-2, development version) laptops from One Laptop Per Child [21].

3. RAINBOW PROTOCOL

The Rainbow protocol is based on two key elements: (1) a mechanism to control access to the shared medium based on each node's content distribution performance, and (2) a network coding scheme for the outgoing data at each node. We detail these two elements in the following subsections.

3.1. Content-directed Medium Access Control

We base Rainbow's MAC on a two-step *innovation reporting* mechanism. First, on each outgoing data packet, a node piggybacks a short report containing the normalized rate at which it received innovative packets—packets carrying new pieces of content—from each neighbor over the past T_r seconds. The innovativeness of packets is determined by the content distribution layer, and so we describe the Rainbow MAC as content-directed. The normalized rate is computed relative to the sending rate, which is derived from sequence numbers; this way, the report implicitly includes the quality of wireless links. To control the size of this piggybacked report, we limit the number of innovation reports that can go into each packet to P. If a node has more than P neighbors to report on, then it will randomly select a subset of size P.

Second, using data from these reports, a node estimates the total *potential innovation* its transmission could provide to neighbors, by summing over all its neighbor nodes, and includes this total in future transmission reports. Nodes then select their transmission rates as follows. A node with the highest potential innovation transmits at the maximum link rate r_L, whereas other nodes send at some small, but non-zero *probing rate* r_P. This way, when the winning node runs out of innovative data to send, other nodes' potential contributions can be gauged from the data sent at the probing rate, and another node promoted to take over the channel.

Under this scheme, each node makes its transmission rate decision based on a comparison of potential contributions of its *one-hop* neighbors; however, the recipients of these contributions reside in the node's *two-hop* neighborhood. As a result, the scheme can properly schedule some difficult topologies such as the cluster-bridge topology from Figure 1; in that case, the bridge's upstream neighbors yield when the bridge receives all available content, and becomes the only useful transmitter among its one-hop neighbors.

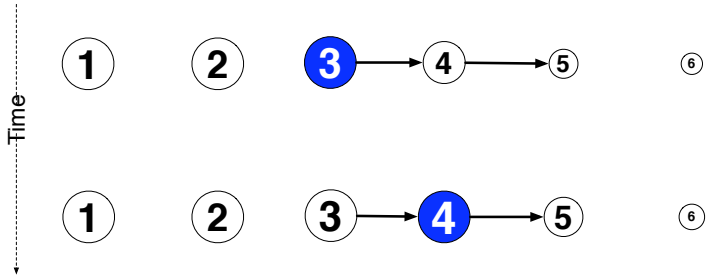

Figure 2. The node contributing innovative packets at the highest normalized rate within a two-hop neighborhood obtains the largest channel share. Assume that content is propagated from node 1 to all other nodes. Size of a node indicates how much innovative content it holds; blue solid fill indicates the node is transmitting at the maximum link rate. At some time point, node 3 delivers innovative content to node 4 at the maximum link rate. Later, node 4 has received the entire content from node 3, dropping node 3's delivery rate of innovative packets to 0. This causes node 4 to be promoted to transmit at the higher rate.

We use a linear topology to illustrate how this scheme naturally handles multi-hop content distribution. As shown in Figure 2, any time when a node at hop i stops being the highest contributor of innovative packets within the one-hop neighborhood, a new high-rate transmitter begins to deliver the content to nodes at hop $i+1$.

3.2. Network Coding Layer

We adopt a random linear coding scheme for network coding, as detailed in [22]. The content to be distributed is assumed to consist of k content symbols $x_i, i \in \{1, \ldots, k\}$, each of which is drawn from a base finite field K, typically $\mathbf{GF}(2^q)$, where q is a small integer such as 8. In coding and decoding, all arithmetic operations performed on symbols are in this base field K.

Under the random linear coding scheme, each transmitted symbol y_i is a random linear combination of ℓ content symbols, i.e., $y_i = \sum_{j=1}^{\ell} c_{i,j} x_j$, where $c_{i,j}$, drawn randomly from the base field K, are *coefficients* of the encoding. A receiver receiving $L \geq \ell$ coded symbols $y_1, \ldots, y_L$ will, with high probability, be able to recover the content symbols $x_1, \ldots, x_\ell$ by solving the following system of linear equations:

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1,\ell} \\ c_{21} & c_{22} & \cdots & c_{2,\ell} \\ \vdots & \vdots & \ddots & \vdots \\ c_{L,1} & c_{L,2} & \cdots & c_{L,\ell} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_\ell \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{pmatrix}$$

To do so, the receivers need the coefficient vector $\mathbf{c_i} = (c_{i,1} \ \ldots \ c_{i,\ell})^T$, so $\mathbf{c_i}$ needs to be transmitted along with the coded symbol y_i. In a packet-switched network, symbols are naturally grouped into packets. To reduce overhead, symbols transmitted in one packet share the same coefficient vector $\mathbf{c_i}$. The overhead incurred by such a random linear coding scheme is thus $\eta = \ell/(m-\ell)$, where m is the maximal number of symbols that a packet can contain, given that most packets are of maximum size in typical content distribution applications.

To manage this overhead when distributing a large amount of content, the entire collection of content symbols is further broken down into *generations*, each consisting of $\ell(m-\ell)$ symbols, or ℓ packets, as suggested in [13]. The total number of generations will thus be $n = \lceil k/(\ell(m-\ell)) \rceil$.

A precode is a fixed-rate forward error correcting code, such as the Reed-Solomon code, with rate λ. This precode is applied to the n generations to give λn *coded* generations. This will alleviate the coupon collector's problem among generations [22].

We store the content in reduced row-echelon form. That is, upon the receipt of a coded packet, a node performs an (incremental) Gauss-Jordan elimination on the *coefficient matrix* $\mathbf{C} = (\mathbf{c_1} \ \mathbf{c_2} \ \ldots \ \mathbf{c_L})^T$. When it terminates, it will report whether this packet has increased the rank of the matrix.

Whenever a node is allowed to transmit, it simply picks uniformly at random a generation among those coded generations of which it has received one or more packets. In theory, it should then form a random linear combination of all the packets it has in that generation. However, since we store the coefficient matrix in reduced echelon form, the node does not need to combine every single packet. Instead, it only picks those corresponding to the non-zero rows in the matrix since the span of these rows already contains all the information that can be derived from the entire collection of received packets in that generation.

The use of network coding is important for content-directed MAC, because it leads to small-sized reports that can piggyback on data packets—a single number, i.e., the rank of a node's coefficient matrix, suffices to describe the "innovativeness" of content held by that node. Furthermore, no additional control messages are needed, such as acknowledgments or descriptions of the content subset held by a particular node.

4. FIELD EXPERIMENT SETUP

4.1. Testbed

We performed outdoor field experiments on a large, flat athletic field on the Harvard campus in Cambridge, MA.

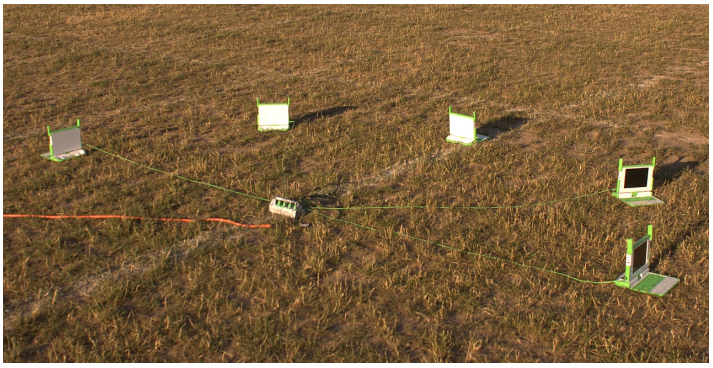

Figure 3. A photo of a 5-node cluster in our outdoor testbed. Clusters of OLPC XO nodes were placed on the ground, and powered externally using the power lines as shown.

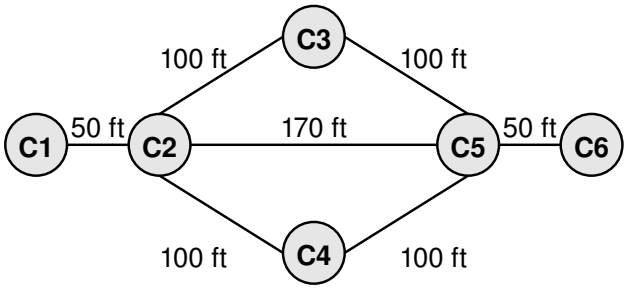

Figure 4. The 6-cluster topology we used in our outdoor experiments, where C_i's denote clusters. Each cluster contains 5 nodes, except the last cluster, which contains 4; for clarity, the individual nodes within clusters are not shown.

Outdoor experiments are attractive because: (1) the amount of external wireless interference (802.11 or any other kind) is relatively small; and (2) results are easier to reproduce in a simpler environment. This is in contrast to the outcome of indoor experiments, which is likely to be dependent on locale-specific factors such as building layout and radio interference conditions at the time of the experiment.

Our testbed consisted of 29 OLPC Beta-2 nodes. These are i386 compatible systems based on the AMD Geode GX processor running at 366MHz, and equipped with 128MB RAM. Each system has one Marvell Libertas 88W8388 802.11b/g radio, with tunable transmit power. All of the nodes were placed on the ground; Figure 3 shows one of the 5-node clusters in our testbed. We used broadcast Ethernet packets at the 2Mbit/s rate for all protocol implementations.

4.2. Network Topology

We focused our experimental work on a class of topologies where nodes reside in clusters. Such topologies are interesting because they require that protocols cope with

		Receiving Cluster					
		C1	C2	C3	C4	C5	C6
Transmitting Cluster	C1	80	47	17	10	0	0
	C2	57	80	35	29	0	0
	C3	2	2	80	37	16	0
	C4	1	5	40	78	8	0
	C5	0	0	7	7	87	71
	C6	0	0	0	0	75	81

Table 1. Average packet delivery probabilities measured between the 6 clusters in our testbed, expressed as percentages. The values on the diagonal are for the links between nodes within single clusters.

the high density of intra-cluster nodes in single collision domains, as well as inter-cluster multi-hop communication.

We ran the majority of experiments on a 6-cluster topology shown in Figure 4; other experiments were run only on clusters 1–3. Nodes in the individual clusters are spaced closely—at most 12 ft apart—so they have almost lossless links to each other. Regardless of the topology, the content source was always a single node residing in cluster 1.

To gauge the link qualities between clusters, we ran a low-rate packet generator on each node and computed the packet delivery probabilities of each sender-receiver pair. These delivery probabilities are summarized in Table 1, as averages of all links between any pair of clusters. These measurements exhibit the complex nature of signal propagation, showing asymmetric delivery rates between pairs of clusters, and imperfect packet delivery even within single clusters. This data will help our understanding of further results obtained on the same testbed.

4.3. Protocols and Performance Metrics

We compare the performance of three flooding protocols for content distribution: Blind (i.e., content-blind), Uncoded, and Coded. The Coded protocol is the Rainbow protocol described in Section 3. We use the other two protocols as points of comparison, and present their design here for completeness.

Blind flooding is a classic flooding protocol enhanced by using a smaller-than-unity relaying probability. Such probabilistic flooding has been proposed in the literature to address the redundancy of broadcast transmissions in the wireless medium [8]. Classic flooding is content-blind, meaning transmission priority is not moderated. In order to use the protocol to send large files the source node splits a file into packet-sized chunks, and floods the chunks sequentially. Once the last chunk is flooded, the source repeats the sequence, starting over from the first chunk.

We have also designed and implemented a content-directed *uncoded* flooding protocol. The basic design is similar to the Selective Forwarding strategy [15], with optimizations specifically designed for wireless networks, and works as follows. The content to be distributed is broken into x fixed-size chunks, each of which consists of y symbols. Every T_b seconds, nodes broadcast an x-bit vector, in which the bits corresponding to the chunks they have are set. By broadcasting its content bit vector, a node implicitly requests missing content from its neighbors. With the knowledge of what neighbors have, a node sorts its chunks according to the *rarity* in the neighborhood, and then transmits starting from the rarest one. One of our optimizations is that nodes yield the medium in the presence of nodes with more content. That is, if a node detects another in its neighborhood with more content than itself, it will transmit at a lower probe rate r_P. Only the node that has the most content in its neighborhood will transmit at full link rate r_L. We refer to this protocol as "Uncoded".

It is important to note that the Uncoded protocol is not simply the Coded protocol without network coding; they have distinct MAC mechanisms chosen to fit the situation where rank information is absent. Recall that the Coded MAC in Rainbow uses historical performance (i.e., innovation reports) over a two-hop neighborhood in deciding whether a node should transmit at the maximum link rate r_L. Network coding enables us to use rank information as a natural way of expressing a transmitter's past performance. Moreover, the stability of a rank-based innovation metric is desirable; good historical performance generally indicates good future performance under network coding. While a similar mechanism in the Uncoded case is also possible in principle, the analogous innovation metric is less stable. For example, under the "rarest first" policy, the node with the best historical performance may abruptly run out of innovative chunks to send; this can lead to thrashing when deciding who should transmit at r_L. To address this problem, our Uncoded MAC makes this decision based simply on the amount of content each node in a one-hop neighborhood possesses. While the Coded and Uncoded MACs are not exactly analogous, the Uncoded protocol is a reasonable content distribution scheme and represents a fair point of comparison.

In the implementation of these three protocols, the neighbor information is timed out if a node has not received any updates from that neighbor for an interval of T_o seconds.

The key metric we use to evaluate performance of the three protocols is the completion time, i.e., the time required for all nodes to obtain the file being distributed. Such a performance metric has also been used by, e.g., [20]. Note that this metric is sensitive to the presence of any poorly performing nodes; for this reason, we limited the experiment running times to a reasonable upper bound that

Name	Description	Value
K	Base field	$\mathbf{GF}(2^8)$
k	Content size	6,120,000 symbols
ℓ	Generation size	100 packets
m	Coded packet size	1,324 symbols
η	Coding overhead	8.2%
n	Number of generations	50
λ	Precode rate	1.1
T_r	Measurement interval	10 seconds
P	Maximum report size	30 neighbors
r_L	Link rate	1.7 Mbits/s
r_P	Probe rate	12 Kbits/s
x	Number of chunks	4,410
y	Uncoded packet size	1,388 symbols
T_b	Announcement period	1 second
T_o	Timeout period	10 seconds

Table 2. Parameters used in the experiments.

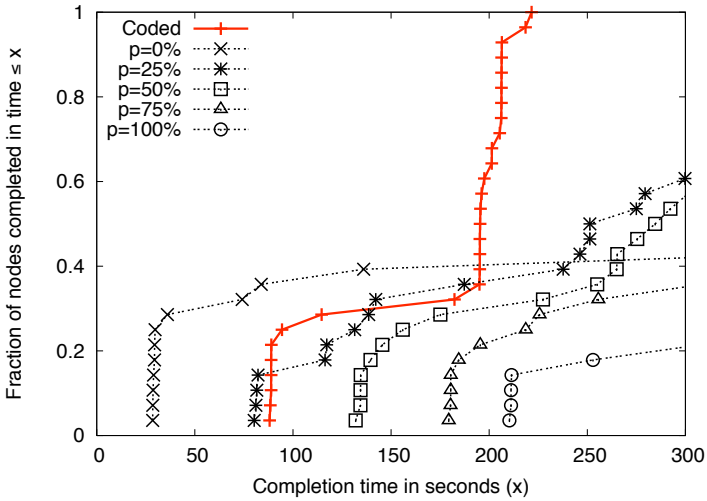

Figure 5. Empirical distribution function of the completion times for single runs of Blind and Coded flooding. The Coded case (i.e., Rainbow) wins in overall completion time. None of the Blind runs completed in the allotted 300s time.

let most nodes complete, yet kept the experiments practical. We have also included several empirical probability distribution functions in the next sections to demonstrate that the completion time can serve as a representative indication of how good a contention distribution protocol is. In particular, we used a small multiple of the content transmission time at link speed. The size of the file we distributed was 6.1 MBytes, which at the 1.7 Mbit/s link rate of our testbed takes about 30 seconds to transfer. We limited the experiment run time to 300 seconds. We summarize additional experiment parameters in Table 2.

5. PERFORMANCE RESULTS AND DISCUSSION

5.1. Comparison of Blind, Uncoded, and Coded Flooding

We report the completion times of the three flooding protocols for content distribution: Blind, Uncoded, and Coded (i.e., Rainbow). The individual node completion times are plotted as empirical distribution functions (EDFs).

Figure 5 shows the performance of five different Blind flooding instances with relay probabilities $p = 0, 25\%, \ldots, 100\%$, as compared to that of Coded flooding (Rainbow). First, note that 0%-flooding completes an initial set of nodes earlier than any other protocol. This behavior shows that simple single-source broadcast can be effective in low packet loss environments with nodes in a single collision domain. However, because relaying is not permitted, nodes out of reach of the source cannot obtain the content. When the relay probability is above zero, we can see that while more nodes get portions of the content, the completion times of the nodes nearest to the source increases. The reason is that channel share is allocated to more nodes than just the source, slowing down the useful transmissions.

Figure 5 also showcases the adaptability of the Coded protocol (Rainbow). That is, Rainbow performs well both in the single collision domain, where its performance is within an order of magnitude of the 0%-flooding, and in the whole topology, where it is the only protocol to complete before the 300s deadline.

Next, we compare the performance of Coded and Uncoded flooding, while at the same time showing that the completion times are stable across different runs. We present this comparison in Figure 6, where we plot the completion distribution functions of three runs each for both Coded and Uncoded. The results show that Coded completes earlier than the Uncoded in all cases. Specifically, when all the nodes in Coded complete, no more than about 60% of nodes in Uncoded have completed.

5.2. Effect of Cluster Size

We ran the Coded and Uncoded protocols on clusters 1–3 of Figure 4, while varying the size of those clusters from 1–5 nodes. We present the results in Figure 7. The key observation we make is that Uncoded performance is significantly degraded with the largest two cluster sizes, 4 and 5. In contrast, the Coded performance remains relatively stable. The key reason for the performance degradation is likely congestion, because unlike Coded, Uncoded allows multiple nodes to transmit at r_L. We provide some qualitative evidence of this in the next subsection.

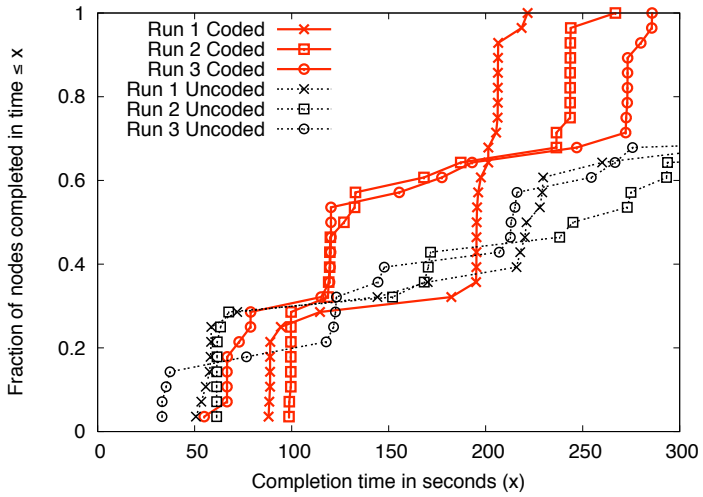

Figure 6. Three runs each of Coded (i.e., Rainbow) and Uncoded flooding show that Coded consistently outperforms Uncoded.

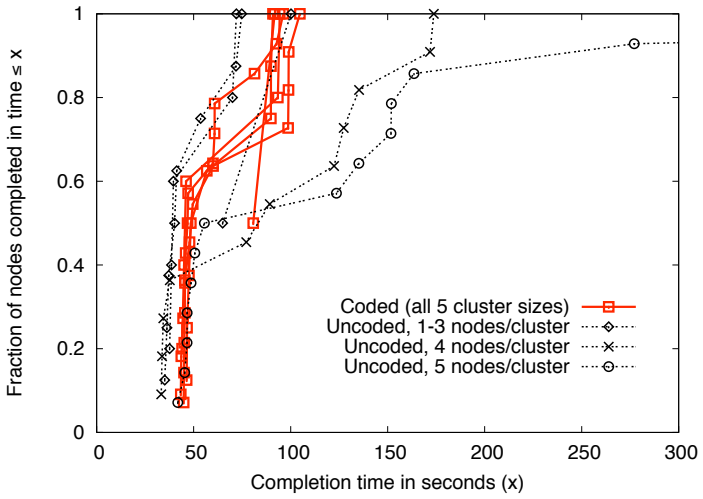

Figure 7. This plot contains the performance results of Coded (i.e., Rainbow) and Uncoded runs on clusters 1–3, while varying the cluster size from 1–5 nodes.

5.3. Dynamic Behavior of Flooding Protocols as Observed by Visualization Tools

We created a set of visualization tools to aid protocol development by inspecting the dynamic behavior of the system. The tools allow us to play back traces collected during experiments and animate several time-varying metrics on a single plot. As an example of how we used the visualizations to gain insight into protocol performance, Figure 8 shows snapshots of Coded and Uncoded runs in the middle of a content distribution, where the content file propagates from the leftmost cluster to the rightmost. In the figures, a node's circle grows as it receives an increasing percentage of the file. The size of a ring around a node's solid circle represents the magnitude of redundant packets received in the past second, where redundant packets are those containing pieces of content already present at a node. When a node transmits at a higher rate, its solid circle has a darker color.

The key metric provided by the snapshots is the transmission rate of the individual nodes at a given time. As we can see, with the Coded protocol only one node transmits at a time; furthermore, the transmitting nodes are activated roughly in a round-robin order, each time delivering a small chunk of the content. In contrast, under the Uncoded protocol multiple nodes transmit at once, leading to collisions at the desired recipients and a slowdown in completion time.

The Coded snapshots provide insight into how Rainbow avoids the bridge lock-out problem. Consider the bottommost Coded snapshot (left panel) corresponding to 100s from start of experiment. The only transmitting node at that time is in cluster 3, acting as a bridge between clusters 1,2 and 5,6. This node is able to transmit because none of its neighbors have been able to deliver innovative packets at a better total rate; therefore, per our MAC scheme in Section 3.1, the other nodes yield.

6. FUTURE WORK

There are several areas of our system where opportunities for more detailed performance analysis exist. While we successfully employed network coding for its qualitative advantages in protocol design, we have not evaluated the quantitative reliability gain suggested by previous work [19] to confirm the results.

The size of coefficient vectors chosen in our current Rainbow implementation was limited by the computation power of the available testbed hardware. At the same time, the maximum size of the transferrable content was dictated by the link layer-defined maximum packet size. Therefore, an open area of future work consists of a thorough exploration of the effect of varying these two parameters.

To our knowledge, no method to estimate the optimal content delivery time, given an arbitrary multi-hop topology with lossy links and the 802.11 protocol model, exists. Such a method would provide a useful lower bound on the achievable completion time for file delivery over real-world systems based on 802.11 radios.

7. CONCLUSION

In this paper, we present Rainbow, a protocol designed for content distribution over multi-hop wireless ad-hoc networks. Rainbow was implemented and validated on a testbed of 29 XO Beta-2 development laptops from One

Laptop Per Child. We compared Rainbow to two other protocols: content-blind probabilistic flooding and content-directed uncoded flooding. As depicted in Figures 5 and 6, Rainbow outperforms probabilistic flooding by 1.3-fold and content-directed uncoded flooding by 1.9-fold, as calculated by the time required for 50% of nodes to reach completion, in the best cases. The reasons for these gains are outlined below.

By virtue of being content-directed, Rainbow is able to optimize medium access control for content delivery, previously impossible under traditional content-blind protocols. The key principle we utilize is that relaying nodes should only occupy the channel under a combination of two conditions: (1) when they have good links to their neighbors and (2) when the innovative content they possess can help the largest number of their neighbors. As a result, under Rainbow, only the nodes that are most capable of delivering innovative content at a given time gain access to the channel. This minimizes exposure to the broadcast storm and bridge lockout problems, and thus makes Rainbow well-suited for multi-hop topologies with bottleneck links, where the lockout problem has greatest impact.

We take advantage of network coding to efficiently implement Rainbow's content-directed MAC. Innovation reports, i.e., those based on rank information, are small enough that they can be piggybacked on coded data packets, allowing nodes running Rainbow to react quickly to transient changes in link quality and content quality of nodes. No separate protocol messages are needed to communicate completion status. Network coding does incur computation overhead at nodes by performing rank calculation and encoding/decoding, as well as communication overhead in transporting coefficients of random linear combinations. The results of this paper suggest that performance gain resulting from using network coding outweighs its overhead in our content-directed MAC.

Finally, we note that Rainbow has the potential to significantly improve applications where reliable data broadcast is required. For example, unmanned aerial vehicles (UAVs) may use Rainbow to distribute surveillance imagery to ground units. In a more sophisticated use case, Rainbow could be used by a UAV to stream surveillance video to ground units participating in a parallelized target discrimination application. That is, Rainbow can play a pivotal role in wireless distributed computing, where it can be considered as a reliable and efficient method for broadcasting the input data of a task to multiple computation nodes. Our recent work [5] in distributed speaker identification over ad-hoc wireless backplanes clearly indicates the need for such a method. Our future work will focus on pursuing this line of investigation.

ACKNOWLEDGEMENTS

We are indebted to One Laptop Per Child for providing us with access to XO Beta-2 development laptops, and Michail Bletsas of OLPC for his comments and encouragement on this project. This research was supported in part by the Air Force Research Laboratory Grants FA8750-08-1-0220, FA8750-08-1-0191 and FA8750-05-1-0035.

REFERENCES

[1] B. Cohen, "Incentives build robustness in bittorrent," 2003.

[2] The annotated gnutella protocol specification v0.4. [Online]. Available: http://rfc-gnutella.sourceforge.net/developer/stable/index.html

[3] I. Stoica, R. Morris, D. Karger, F. M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM*, 2001.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, 2001.

[5] H. T. Kung, C.-K. Lin, C.-Y. Su, D. Vlah, J. Grieco, M. Huggins, and B. Suter, "A computational wireless network backplane: Performance in a distributed speaker identification application," in *MILCOM*, 2008.

[6] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, 2000.

[7] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," in *DIALM*, 1999.

[8] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MOBICOM*, 1999.

[9] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, 2000.

[10] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz, "Unfairness and capture behaviour in 802.11 adhoc networks," in *Communications, IEEE International Conference on*, 2000.

[11] F. Togabi and L. Kleinrock, "Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy tone solution," *Communications, IEEE Transactions on*, vol. 23, no. 12, pp. 1417–1433, December 1975.

[12] OPNET Inc., "OPNET Modeler version 14.0. Discrete Event Simulation API Reference Manual," http://www.opnet.com.

[13] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference on Communication, Control, and Computing*, 2003.

[14] M. Mitzenmacher and E. Upfal, *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.

[15] A. A. Hamra, C. Barakat, and T. Turletti, "Network coding for wireless mesh networks: A case study," in *WOWMOM*, 2006.

[16] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla, "Code torrent: content distribution using network coding in vanet," in *MobiShare*, 2006.

[17] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[18] M. Wang and B. Li, "Lava: A reality check of network coding in peer-to-peer live streaming," in *INFOCOM*, 2007.

[19] M. Ghaderi, D. Towsley, and J. Kurose, "Network coding performance for reliable multicast," in *MILCOM*, 2007.

[20] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *INFOCOM*, 2005.

[21] One laptop per child. [Online]. Available: http://laptop.org/laptop/hardware/specs.shtml

[22] P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Annual Allerton Conference on Communication, Control, and Computing*, 2006.

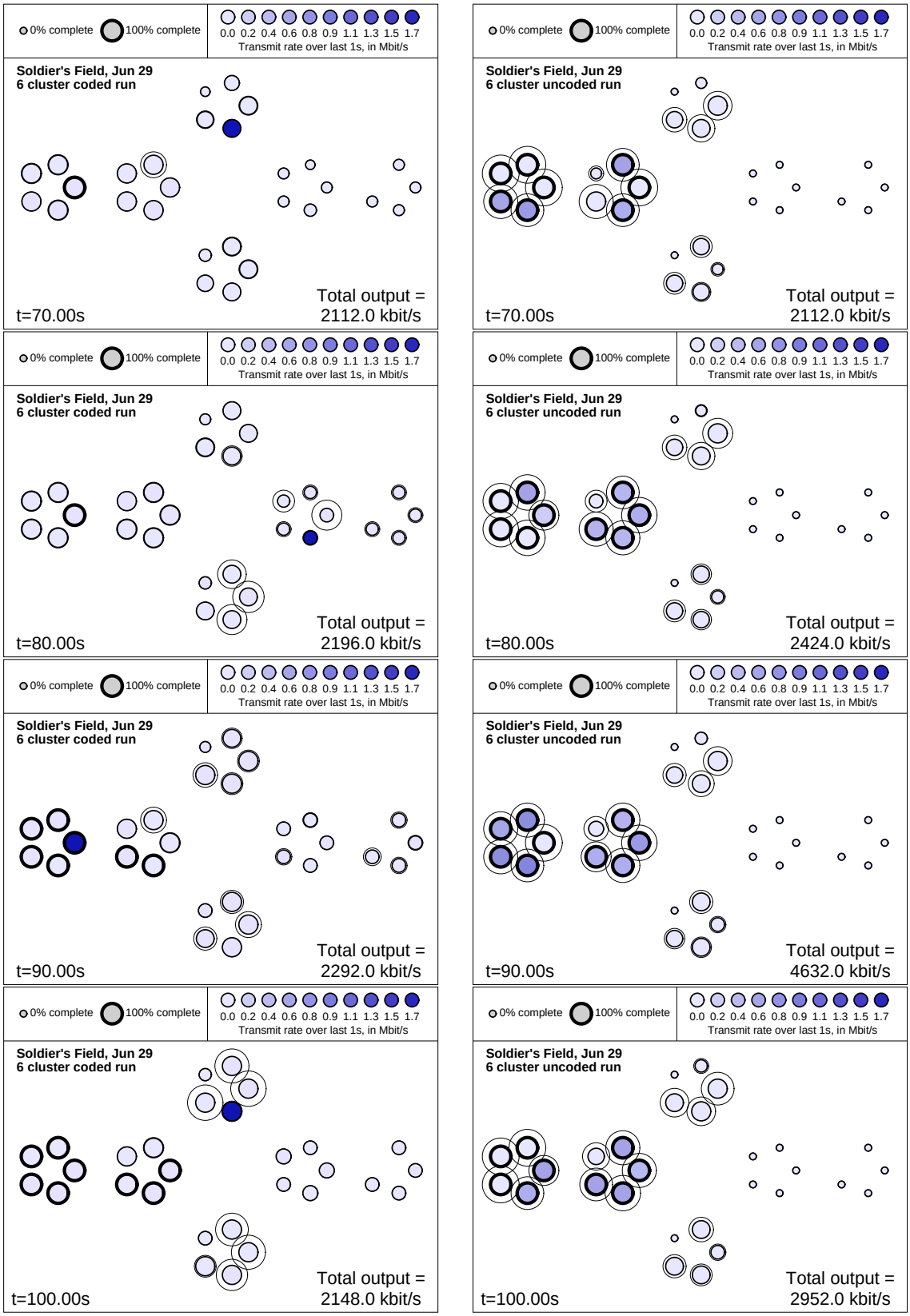

Figure 8. Snapshots of Coded (left panel) and Uncoded (right panel) content distribution protocols at time points 70 – 100s, 10 seconds apart. Under Coded distribution, only one sender tends to transmit at a high rate, while under Uncoded, several nodes are active at once, causing collisions at receiving nodes. As a result, the Uncoded protocol is less effective at delivering data to the right-most clusters.