# From Rateless to Hopless

Zhenjiang Li[1], Wan Du[1], Yuanqing Zheng[2], Mo Li[1], Dapeng Wu[3]
[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]Department of Computing, Hong Kong Polytechnic University, Hong Kong
[3]Department of Electrical & Computer Engineering, University of Florida, USA
{lzjiang, duwan, limo}@ntu.edu.sg, csyqzheng@comp.polyu.edu.hk, wu@ece.ufl.edu

## ABSTRACT

This paper presents a hopless networking paradigm. Incorporating recent techniques of rateless codes, senders break packets into rateless information streams and each single stream automatically adapts to diverse channel qualities at all potential receivers, regardless of their hop distances. The receivers are capable of accumulating rateless information pieces from different senders and jointly decoding the packet, largely improving throughput. We develop a practical protocol, called HOPE, which instantiates the hopless networking paradigm. Compared with the existing opportunistic routing protocol family, HOPE best exploits the wireless channel diversity and takes full advantage of the wireless broadcast effect. HOPE incurs minimum protocol overhead and serves general networking applications. We extensively evaluate the performance of HOPE with indoor network traces collected from USRP N210s and Intel 5300 NICs. The results show that HOPE achieves 1.7x and 1.3x goodput gain over ExOR and MIXIT, respectively.

## Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Network Protocols**]: Routing protocols

## Keywords

Rateless codes; wireless networks; routing

## 1. INTRODUCTION

Conventional multi-hop wireless networks route packets hop-by-hop along a path of favorable links from the source to the destination [1, 22, 6], e.g., $s \to n_1 \to n_2 \to d$ in Fig. 1(a). The validity of multi-hop routing is based on the belief that any other paths cannot provide higher throughput than the selected one. Instead of choosing a fixed path, *opportunistic routing* approaches delay the path selection and broadcast packets to a sub-set of nodes at each hop, exploiting the opportunities of the successful packet reception over the traditionally unfavorable links [2, 3, 14], e.g., $s \to n_2$, $s \to d$, and $n_1 \to d$ in Fig. 1(b). Nevertheless, wireless channels are still under utilized as opportunistic routing is usually packet-oriented, which turns only the intact packet opportunities into effective throughput.

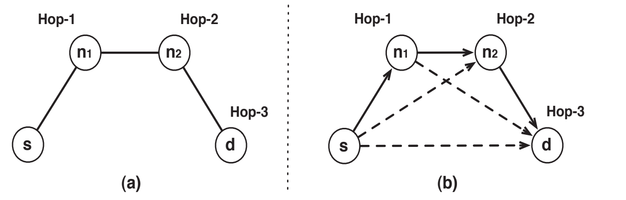

**Figure 1: (a) A three-hop path in a wireless network. (b) Favorable and unfavorable links in the network.**

This paper studies from an information-oriented perspective and observes that while the data packet is delivered in a hop-by-hop manner, e.g., $s$ transmits packets to $n_1$ in Fig. 1(b), the information contained in the transmission is spread beyond each hop. All the network nodes are potential receivers and they have the opportunities to receive different amount of information from the sender, though the information may not be immediately adequate to be translated to an intact packet. In this paper, we apply recent rateless coding schemes and encode packets into rateless information streams. Rateless codes adapt each individual data transmission to multiple receivers, automatically synchronizing the effective data rate to the diverse channel qualities of all the wireless links. At the receiver side, channels from all the potential senders are viewed as a single joint information channel. Each node accumulates rateless information from the joint information channel regardless of hop distances, and jointly decodes the packet using the information pieces from multiple senders. We name such a networking paradigm *hopless networking*. Hopless networking breaks the hop-by-hop packet transmission into the hopless information accumulation. The applied rateless codes allow each node to best adapt to the information quality.

We develop a hopless data forwarding protocol, HOPE, to instantiate the hopless networking paradigm. HOPE entails a non-trivial design which concerns forward scheduling, role transition of nodes, protocol stack development, etc. At any time, HOPE selects a "best" candidate from the nodes possessing a packet to forward it in the network until another "better" candidate decodes the packet. To do so, HOPE prioritizes intermediate nodes by their channel qualities to the destination, and performs a distributed scheduling to delegate the forwarding "right" among the nodes. Each node switches its role between the information accumulator, delegated source, and destination in forwarding the data packet. All above operational logics are assembled in a hopless layer, which works with the link and network layers in the traditional protocol stack. The hopless layer can be further extended down to work with the recent physical layer rateless codes for higher throughput. We carefully address the practical challenges to incorporate the existing rateless codes with multiple senders and receivers in the hopless data packet forwarding. The joint rateless coding scheme efficiently aggregates information from multiple senders and decodes the original data packet without a central coordination. Compared with the existing opportunistic routing family, HOPE has the following advantages.

- HOPE accumulates the rateless information from multiple senders, and jointly decodes the packet to improve throughput over the packet-oriented approaches [1, 2, 3, 14]. In HOPE, the successful packet decoding and forwarding at all potential receivers are essentially asynchronous, involving small coordination overhead.
- Data forwarding in HOPE best adapts to the channel quality diversity and makes full use of the wireless broadcast effect. Recent studies (e.g., SPaC [9] and MIXIT [20]) cannot fit all potential receivers in one transmission and are thus suboptimal in utilizing the hopless information.
- Unlike the existing flow-based approaches built on intra-flow network coding (e.g., MIXIT [20], MORE [3], FUN [16]), HOPE does not feature the aggressive usage of the network bandwidth. HOPE smoothly works with the conventional CSMA, and performs well with the flow traffics and the intermittent traffics in the network.

We extensively evaluate HOPE with the indoor network traces collected from 19 USRP N210s and 19 Intel 5300 NICs. We compare HOPE with state-of-the-art opportunistic routing approaches. The results show that compared with ExOR [2] and MIXIT [20], HOPE improves average goodput by 1.7x and 1.3x respectively in both single- and multiple-flow scenarios.

In the rest of this paper: we introduce the motivation and the hopless networking in § 2. § 3 details the design of HOPE. In § 4, we conduct extensive performance evaluation. § 5 reviews related works and § 6 concludes this paper.

## 2. MOTIVATION

### 2.1 Packet-oriented v.s. information-oriented

*Packet-oriented*: We deploy 4 USRP software radios based on the topology of Fig. 1 and look into the detailed channel qualities when packets traverse from $s$ to $d$. The USRP nodes are configured to operate on OFDM at 2.4GHz. The detailed experiment settings will be given in §7. We let $s$, $n_1$, and $n_2$ send 40 packets in sequence and measure the SNRs at downstream nodes. Fig. 2 depicts the measured SNRs of the packets "heard" over all 6 links in Fig. 1(b). The favorable links, $s \rightarrow n_1$, $n_1 \rightarrow n_2$, and $n_2 \rightarrow d$, selected in conventional multi-hop networks, experience good average SNR

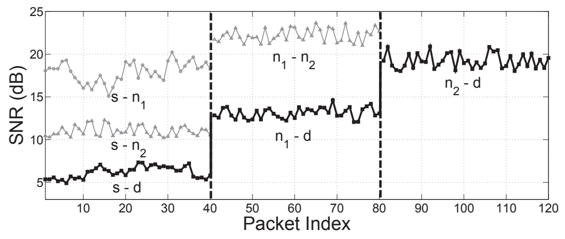

Figure 2: Channel SNRs across different links.

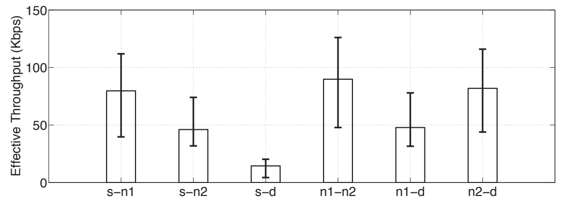

Figure 3: Effective throughput across different links.

($\geq$15dB). The traditionally unfavorable links, $s \rightarrow n_2$, $s \rightarrow d$, and $n_1 \rightarrow d$, experience poorer SNR conditions. In conventional multi-hop networks, the information from unfavorable links is usually not utilized if it is not adequate to translate to a decoded packet. Being packet-oriented, the multi-hop networking keeps routing intact packets to a "better" relay at each hop until the packets reach the destination. Benefitting from the wireless broadcast effect, however, when one node sends a packet, all nodes have access to the transmitted signals though with distinct qualities.

*Information-oriented*: This paper studies from an information oriented perspective and aims at improving throughput with the previously underutilized information across multiple hops (which we call hopless information). We apply the recent rateless codes to encode the raw data packets into a stream of rateless units. Rateless codes allow the sender to incrementally reduce the effective data rate by sending more rateless units, which adaptively matches the channel condition to the receiver.

We experiment with the USRP testbed and quantify the potential gain of making use of the hopless information. We leverage Spinal codes [26] to perform rateless coding on the raw data packet. Spinal codes apply a pseudo-random hash function to the raw data bits to produce a sequence of coded symbols in a way that "*every achievable higher coding rate is a prefix of achievable lower rates*" [26]. A constant number of coded symbols are packed together to form each rateless unit, named Spinal "pass". Spinal codes allow a sender to gradually reduce the effective data rate by sending more Spinal "passes".

We let $s$ keep transmitting Spinal "passes" for a $L$-bit packet. The three nodes $n_1$, $n_2$, and $d$ simultaneously receive the "passes" from $s$. As the channel quality of $s \rightarrow n_1$ is better than $s \rightarrow n_2$ and $s \rightarrow d$, we expect that $n_1$ first decodes the packet with fewer "passes". During the time, $n_2$ and $d$ accumulate the same number of "passes", but containing more symbol errors. We keep $s$ sending the "passes" until $n_2$ and $d$ are eventually able to decode the packet. We record the time $t$ of each node in decoding the $L$-bit packet and derive the *effective throughput* $etp = L/t$. We repeat the experiment 100 times and obtain the effective throughput achieved on all $s \to n_1$, $s \to n_2$, and $s \to d$. We also do the same experiment for the rest 3 links. In Fig. 3, we observe that the effective throughputs on traditionally unfavorable links $s \to n_2$ and $s \to d$ are substantial, i.e., when $n_1$ decodes a packet from $s$, the hopless information accumulated at $n_2$ and $d$ statistically corresponds to 56% and 18% of the packet. Link $n_1 \to d$ also has substantial effective throughput, e.g., 53% of $n_1 \to n_2$'s.

## 2.2 Hopless networking paradigm

We propose a hopless networking paradigm to incorporate the potential effective throughput gained from the hopless information. In hopless networking, a receiver decodes the packet using the rateless codes from multiple senders. From the receiver's point of view, however, it is unnecessary to identify the information pieces from each individual sender. Instead, the receiver may treat the entire network as a whole and view the channels from each sender as a single information channel, which is called *joint information channel* in this paper. The receiver then takes information from its joint information channel. The highlighted links $s \to d$, $n_1 \to d$ and $n_2 \to d$ in Fig. 2 compose the joint information channel for $d$. With the help of rateless codes, $d$ is able to make the full use of hopless information from both the conventional favorable links $n_2 \to d$, and the complementary links $s \to d$ and $n_1 \to d$.

Fig. 4 depicts the general hopless networking paradigm, where each node $n_i$ is connected to the network through its joint information channel. As adequate information is accumulated from the network, $n_i$ is able to decode the packet. At this time, if $n_i$ is the "best" candidate to forward the packet, it takes the place of the packet forwarding and injects the rateless codes of this packet to the network until another "better" candidate decodes it. With such a networking paradigm, whenever the information of a fresh packet is available in the network, any node can fully accumulate it through its own joint information channel. On the other hand, when any node transmits a packet, the rateless codes automatically adapt to the channel qualities of all potential receivers (explained below).

A natural question one may ask is: *Why the potential throughput gain revealed in hopless networking has kept being overlooked in previous multi-hop networks?*

Many existing studies tried to harvest such potential gains by aggressively capturing opportunistic packets from unfavorable links. Existing packet-oriented opportunistic routing schemes, e.g., ExOR [2] and MORE [3], however, cannot make the full use of those low-quality links. Only successfully decoded packets surviving from low quality links are utilized.

Some latest works [9, 20] notice similar gain of exploiting partial information from corrupted packets and value the "clean" bits or symbols from corrupted packets. By packet combinations or network coding schemes, they can make a better use of the low-quality links. Constrained by the unified data rate of each transmission from the sender, those works still cannot make the full advantage of wireless broadcast effect. For the example in Fig. 2, $s$ usually sets a suitable data rate that matches the per-hop link quality of $s \to n_1$. The high data rate for the good link, however, would be too high for weak links like $s \to n_2$ and $s \to d$ to accept enough correct symbols or data bits. Thus, only when the weak links dramatically improve can intact packets and yield the opportunistic throughput gain. A low data rate helps to increase the opportunistic gain over weak links, but on the other hand, limits the throughput that can be achieved over good links. Similar problems also exist on $n_1 \to n_2$. In conventional multi-hop networks, a sender always faces the dilemma of adapting to an impossible best sending rate, which has long been considered an open problem in opportunistic routing [2, 3].

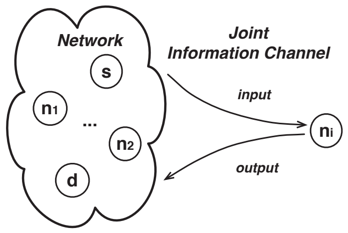

**Figure 4: Hopless networking paradigm for an arbitrary node $n_i$.**

Hopless networking hence outperforms prior designs from two aspects: 1) Each *receiver* can adapt to the joint information channel and the success of packet decoding attributes to the rateless information from multiple senders; 2) Each *sender* can adapt to the joint information channel and produce rateless data streams that automatically adapt to all various links.

## 3. DESIGN

We instantiate the hopless networking paradigm and develop HOPE, a non-trivial design to coordinate the network and acquire full throughput gain in the hopless forwarding.

## 3.1 Design principle

In HOPE, each packet is rateless coded into a stream of units. Many existing rateless codes can be applied, including LT [24] and Raptor codes [30] for the link layer bit blocks or recent Strider [10] and Spinal codes [26] for the physical layer symbols. For each packet delivery, all nodes in the network play the roles of the source, the accumulator, or the destination to forward data. We elaborate the different roles and their interactions in HOPE. We aim at giving a high level behavior description and temporarily omit the technical details for the sake of a clear presentation. Details are introduced in §3.2.

**The accumulator**: In HOPE, all nodes work as accumulators. An accumulator listens to the network and accumulates rateless units for each undecoded packet until the packet can be decoded or the decoding is aborted.

**The source**: There are two types of sources in HOPE: *the original source* and *the delegated source*.

The original source initiates the packet transfer and sends out the stream of rateless data units. The source keeps sending until the packet gets ACKed.

A delegated source comes from an accumulator that successfully decodes a packet and is ready to forward it. Multiple accumulators may transit to delegated sources concurrently for the same packet. A priority list specifies the priority rank of all the accumulators, and HOPE leverages a priority based back-off scheme (§3.3) such that the delegated source with the highest priority is more likely to forward first

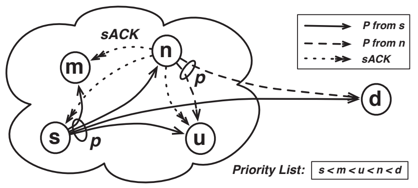

**Figure 5: An example of packet transmissions.**

and the potential duplicated forwardings from new delegated sources with lower priorities are prevented.

**The destination**: The destination behaves the same as an accumulator except that it does not further forward packets.

After decoding a packet, the destination or the new delegated source sends out an ACK (via broadcast) to stop the previous delegated source's transmission. The ACK contains the IDs of the ACKed packet (e.g., $p$) and the delegated source (e.g., $n$), which can also make accumulators with lower priorities than $n$ abort the decoding of packet $p$ (as a higher priority node $n$ has already decoded $p$). In case of multiple delegated sources for the same packet, the ACK sent from a high priority delegated source prevents the duplicated forwarding from the lower priority delegated sources. Different from traditional per-hop ACKs, the purpose of ACKs in HOPE is to switch and control the data forwarding roles, so we denote them as sACKs.

Fig. 5 illustrates the interaction of different roles with an example scenario. The original source $s$ starts sending a packet $p$ to destination $d$. At the beginning, nodes $n$, $m$, $u$, and $d$ all work as accumulators, among which nodes $n$ and $u$ first successfully decode the packet and become the delegated sources. Due to the priority based back-off (§3.3), node $n$ sends out an sACK to stop $s$ and $u$ from forwarding, and aborts the decoding of packet $p$ at $m$. On the other hand, $d$ keeps accumulating the rateless units from $n$ and tries to decode the packet with the stored rateless units from both $s$ and $n$.

## 3.2 Hopless Layer

Although the design emphasis of HOPE is fundamentally different from the existing hop based paradigm, a complete redesign of existing protocol stack is less attractive. We seek to provide a clean abstraction of the hopless operations and integrate it with the current protocol stack. Fig. 6 depicts the architecture of HOPE. HOPE adds a *Hopless Layer*, primarily working with the link layer as follows.

To transmit a packet, the data payload is first delivered from upper layers to the *Role Transition Module* of the hopless Layer. The role transition module locally decides whether the transmission should be aborted if other higher priority nodes have already transmitted this packet to the network. The decision is made based on the *Priority list* and the *Packet list*. If the transmission can continue, the *Information Bucket* generates rateless units of the packet, and adds a hopless header in front of the rateless units. The rateless units with the hopless header are finally passed to the link layer and undergo CSMA before the transmission.

For receiving, the PHY layer loads the bits from a receiving buffer after detecting a preamble. It passes the bits to the link layer, which are further retrieved by the information bucket to perform the rateless decoding. In addition,

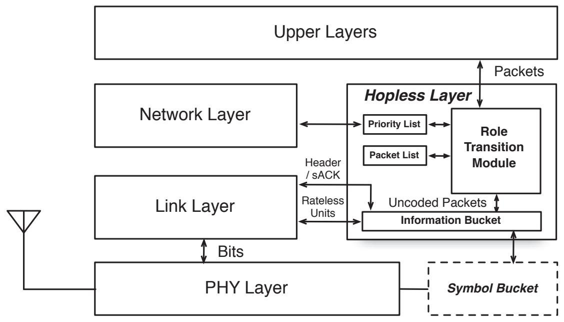

**Figure 6: HOPE architecture.**

the packet list is updated based on the hopless header of the received packet. If the rateless code requires to access PHY symbols, we extend the information bucket to the PHY layer and introduce a *Symbol Bucket* to substitute the function of the information bucket.

In the rest of this section, we present the detailed design of the hopless layer.

**Information bucket**. The hopless layer receives rateless units from the link layer. An information bucket is maintained to organize rateless units of all undecoded packets. For an incoming data packet $p$, a node $n$ retrieves the original source, destination, and sequence number from its header, which uniquely identifies this packet. If the packet has not been locally decoded before (by looking up the packet list which will be detailed later), the node registers a unique piece of buffer to accumulate the rateless units of the packet. If $p$ has already been registered, the rateless units are directed to the existing buffer of $p$. Node $n$ performs rateless decoding for the packet whenever new rateless units are received. If decoding is successful, the decoded packet is passed to the state transition module and the buffer is released. If decoding fails, the buffer is maintained to accumulate more rateless units. The main functionality of the information bucket is to incorporate the rateless coding scheme in receiving packets from the joint information channel. The partial information of undecoded packets is stored orderly until the packets get decoded.

The information bucket can be extended down to work with the PHY layer. A symbol bucket can be included, which directly fetches PHY symbols and can perform recent physical rateless coding schemes. In our current design, we use Spinal codes as the default scheme. The symbol bucket accumulates the Spinal "passes" of symbols and performs rateless decoding for each packet. The symbol bucket substitutes the information bucket and preserves all operation logics. By performing the physical layer rateless codes, the capability of the information accumulation improves.

**Priority list**. A priority list is maintained for each destination in the network, which evaluates the data forwarding utilities of all nodes to the destination, based on which their forwarding priorities are ranked. Traditional routing metrics, e.g., the geographic distance [19], ETX [5], etc, characterize the packet-oriented transmissions, which do not truthfully evaluate the forwarding utilities atop the hopless networking. To this end, in HOPE, we propose the end-to-end effective throughput from the current node to the destination as the cost metric. A packet may be forwarded by arbitrary delegated sources from the current node $n$ to the destination $d$. Suppose the packet length is $L$. The effective throughput from $n$ to $d$, denoted as $etp_{n,d}$, can be approximated as:

$$etp_{n,d} = \max_{i \in N}\{\frac{L}{t_{n,i} + (L - t_{n,i} \times \frac{L}{t_{n,d}})/etp_{i,d}}\}, \qquad (1)$$

where $t_{i,j}$ is the time required for node $i$ to deliver a packet to node $j$ with the best data rate, and $N$ is the set of all nodes in the network. The variables $t_{n,i}$ and $t_{n,d}$ in Eq. (1) can be directly measured by $n$ with probing packets. Probing packets are decoded directly from one sender each time, without the accumulation from other nodes. The $etp_{n,d}$ can thus be derived through the periodic channel estimation and iterative state exchange among nodes, similar to most existing routing protocols. Therefore, we reuse the original periodic channel estimation module in the network layer, but intercept probing packets and add the hopless header before releasing them to the link layer. Similar to existing routing protocols, e.g., [5, 31, 15, 25], the frequency to probe channels is set to be once every tens of minutes.

Based on the channel probing, the priority list for each destination $d$ is then formed and maintained based on the descending rank of $etp_{i,d}$. As all the nodes in the network are ordered in the priority list, the conventional routing loop (oscillation between different delegated sources) will not occur in HOPE. In addition, HOPE also tolerates certain suboptimal selection of the delegated sources from the priority list because its rateless feature. Therefore, each node reports their updated $etp_{i,d}$ infrequently to reduce the communication overhead[1], e.g., with a long period length, or after the transmissions of several data batches from the source node. After collecting $etp_{i,d}$ from each node, the destination reorders the nodes in the network and distributes the updated priority list to all the nodes using the standard flooding technique. In our current implementation, the priority list is empirically updated every 30 minutes. Between two consecutive updates, the priority lists of each node are not changed.

**Packet list**. Each node in HOPE also maintains a packet list that reflects the packet possessions in the network, based on which the node can locally manage its actions and role transitions. An entry is maintained in the packet list for each packet according to its source ID, destination ID, and sequence number. Each packet entry records the highest priority node (based on the node's best local knowledge) that possesses the intact packet. Each node dynamically updates the packet list upon the reception of a data packet or an sACK. If the decoded packet or the sACKed packet is not in the list, the node creates a new entry. A packet entry thus does not necessarily indicate the packet decoded by the node. The node may only receive sACKs about this packet. Packet entries are periodically flushed to control the storage overhead. We adopt the same period length as the priority list update to flush the packet list since we do not observe the sojourn time of one data file packet greater than 30 minutes in our study.

**Role transition**. In HOPE, nodes transit between two roles, accumulator and delegated source. Initially, the original source directly acts as a delegated source and all other nodes start from accumulators. Destination is a special accumulator.

Being an accumulator, a node (e.g., $n$) waits for incoming units. The node tells different unit types based on a field in the header (UNIT_TYPE detailed in Fig. 7). If the incoming unit is an sACK, node $n$ checks the entry about the sACKed packet in its packet list. If the recorded packet possessor in the list has lower priority, node $n$ updates the entry with the possessor indicated in the sACK. If no entry is found about the sACKed packet, a new entry is created. A node remains as an accumulator after receiving an sACK. If the incoming unit is a rateless unit, node $n$ accepts it in its information bucket. However, prior to decoding, the node refers to the packet and priority lists and takes the following possible actions:

*Role transition*: If the rateless unit sender has lower priority than node $n$, and a) the packet is not in the packet list or b) the packet is in the list but the recorded possessor has lower priority than node $n$, node $n$ accumulates the rateless unit and performs rateless decoding in the information bucket. In case the decoding is successful, node $n$ transits to a delegated source and updates the packet entry in the packet list with its own ID (If no such an entry, the node creates one); otherwise it remains as an accumulator and waits for new incoming units. Becoming a delegated source, node $n$ first broadcasts an sACK after wining the channel contention. Prior to transmit rateless units, the node further refers to the packet and priority lists. If the packet list indicates that the data packet is owned by some higher priority node, the delegated source cancels the packet forwarding and then transits back to the accumulator (it happens when multiple accumulators become delegated sources concurrently and node $n$ receives an sACK from a higher priority node); Otherwise, node $n$ generates a stream of rateless units and sends them out. After sending a rateless unit, node $n$ transits back to an accumulator for receiving possible sACKs. If no acknowledgement is received within a time interval, node $n$ turns to the delegated source again and sends out another stream of rateless units.

*sACK transmission without role transition*: If node $n$ has higher priority than the sender of the incoming rateless unit and has already decoded this packet, it transmits an sACK and stays as an accumulator. Another possibility of merely transmitting an sACK without role transition is that node $n$ has not decoded the packet, but the packet entry already exists (due to a previous sACK). In this case, if the recorded possessor in the packet list has higher priority than the sender, node $n$ transmits an sACK as well. The sACK is to inform the sender to stop transmitting and update its packet list.

*No role transition and communication*: If the sender of the incoming unit has higher priority than node $n$, node $n$ takes no communication actions. It updates its packet list, if the sender has higher priority than the recorded one.

In summary, the knowledge about the packet possession is mainly updated through sACKs. An sACK is sent out when either a node decodes a new packet or detects a packet already decoded by a higher priority node while still transmitted by a lower priority node so as to prevent such an unnecessary forwarding. Each sACK is broadcasted once and not all nodes are ensured to receive it. As a result, multiple delegated sources may transmit the same packet concurrently. Since the knowledge about the packet possession is updated

[1]In HOPE, nodes explicitly report $etp_{i,d}$, instead of including it in data packets, because some nodes may not participate in the data delivery from a source to a destination.

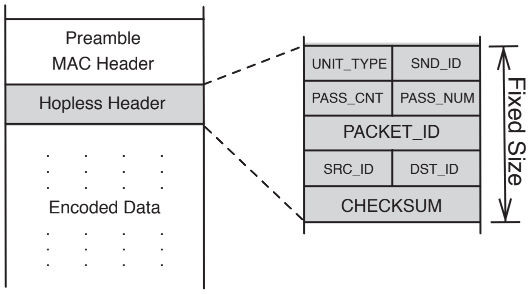

**Figure 7: The format of HOPE units.**

through sACKs as well as rateless units, a delegated source can immediately cancel the packet forwarding once it knows a higher priority node that possess this packet. HOPE tolerates such inconsistency. In §4, we experimentally examine the overhead due to duplicated transmissions and the result shows the cost is generally small, e.g., $< 10\%$ of the total throughput.

## 3.3 Practical Issues

To translate the HOPE design into a working protocol, we need to carefully address several practical issues, which are discussed in the following.

**Design and format of rateless units**. HOPE supplements a fixed length header for each rateless unit. Fig. 7 depicts the header format. UNIT_TYPE tells the current unit is a rateless data unit, sACK, or probing packet. The ID of the current sender is stored in the SND_ID field. The packet ID (PACKET_ID), the source ID (SRC_ID), and the destination ID (DST_ID), uniquely identify the packet that the current rateless unit belongs to. The PASS_CNT and PASS_NUM fields are two parameters related to the Spinal codes which will be explained later. A CHECK_SUM field is applied to protect the header. The total overhead of header is only 12 bytes in our current design.

In HOPE, the header contains meta information to distinguish different units. The rateless units of the same packet transmitted by different delegated sources have different headers, so the headers are not rateless coded. Following the design in 802.11, HOPE lets nodes transmit plain headers with the lowest bit rate (e.g., $6.5Mbps$ in 802.11n networks [34]) so as to lower the SNR requirement in acquiring the headers. This maximizes the range of candidate accumulators that can successfully retrieve the headers and accumulate the rateless units. According to our experimental results in §4, the slow rate header transmission can be decoded with as low as 5dB SNR. sACK is a special unit with preamble [13, 12, 29] and header only.

**Rateless decoding from joint information channel**. None of existing rateless codes consider jointly decoding data from different senders with different data qualities. HOPE extends Spinal codes to decode data "passes" from multiple senders, which can be similarly made to other rateless codes.

*Pass numbering*: In HOPE, the Spinal codes hash function $H$ used across all nodes is the same. To maximize the information gain from the accumulated "passes", different delegated sources forward different "passes" for the same data packet. To make "passes" independent with each other, a delegated source transmits symbols following the latest "pass" received from the previous delegated source, e.g., if the latest "pass" received by node $n_2$ from $n_1$ is $pass_1$, $n_2$ can start transmitting from $pass_1 + 1$. To add some tolerance in accommodating more "passes" ($n_1$ may continue transmitting extra "passes" until sACKed, and those "passes" may be received and made use of by some accumulators), $n_2$ may skip a random number $b$ "passes" and start from $pass_1 + b$. The start point is specified by PASS_NUM in the header for the receivers' decoding. On the other hand, each rateless unit can contain fixed number of Spinal "passes", e.g., PASS_CNT=1 in Fig. 7. It can also be set adaptively [23].

*Weighted decoding*: After a delegated source transmits the $j$th "pass" including $N$ symbols, $\mathbf{X}_j(M) = [x_1, x_2, \ldots, x_N]$, for a data packet $M$, we denote received symbols as $\mathbf{Y}_j = [y_1, y_2, \ldots, y_N]$. In AWGN channels, a max-likelihood decoder requires to satisfy:

$$\max_M\{p(\mathbf{Y}_j|\mathbf{X}_j(M))\} = \max_M\{\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(y_i-x_i)^2}{2\sigma_{ij}^2}}\},$$

where $\sigma_{ij}^2$ is the noise variance of the $i$th symbol in the $j$th "pass". Taking the natural logs and dropping terms that are not a function of $M$, we obtain:

$$\max_M\{\ln p(\mathbf{Y}_j|\mathbf{X}_j(M))\} \propto \min_M\{\sum_{i=1}^{N} \frac{(y_i-x_i)^2}{\sigma_{ij}^2}\}. \tag{2}$$

To maximize the decoding probability, Eq. (2) implies that the "passes" with lower SNRs should contribute less to decoding compared with those with higher SNRs. In the original Spinal codes, received symbols are viewed equally in decoding, as they are from the same sender over a relatively stable channel. We extend the Spinal decoder to $\min_M\{\sum_{i=1}^{N} \frac{(y_i-x_i)^2}{\sigma_j^2}\}$, where $\sigma_j^2$ is the SNR of the $j$th "pass" measured from the rateless unit preamble. One may further supplement a packet postamble to improve the SNR estimation as [18], e.g., measuring $\sigma_j$ with a higher accuracy from both the preamble and the postamble.

**Multiple access**. In HOPE, each sender strictly follows CSMA for channel access. When accumulating information from one source, an accumulator always launches the preamble detection when it detects a sudden increase of RSS (Received Signal Strength). This is because the sudden increase of RSS usually indicates a new incoming unit with higher SNR and it is more beneficial for the receiver to accumulate the high-SNR information. The preambles or headers of the incoming unit may not be correctly detected if its RSS is not strong enough. In such a case, traditional networks cannot hook on the incoming data as well. Traditional packet collisions at the receiver translate to very low SNR rateless units received in HOPE, which are usually assigned negligible weights in the weighted decoding scheme in Eq. (2).

Multiple nodes may decode a same packet and send out sACKs simultaneously. In this case, collision occurs and the sender fails to be notified. To address such an issue, receivers conduct a priority based back-off before sending the sACK. Higher priority nodes are prone to have shorter back-off delays. In HOPE, the priority list is equally divided into $m$ groups (the last group may contain fewer nodes). Nodes in group $i$ performs back-off in $[0, 2^i - 1] \cdot 16\mu s$, where $16\mu s$ is the *SIFS* duration. Thus, the time interval for the delegated source to transmit the next round of rateless units equals to the maximum back-off latency plus an sACK transmission delay. In §4, the maximum back-off latency is set to $128\mu s$.

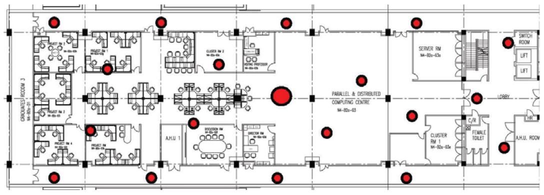

**Figure 8: Traces are collected from 19 positions in a 800$m^2$ office room.**

## 4. PERFORMANCE EVALUATION

We compare HOPE with: a EXT-based Single Path Routing protocol, SPR [5]; a classical opportunistic routing protocol, ExOR [2]; and a state-of-the-art symbol-level network coding based opportunistic routing protocol, MIXIT [20].

### 4.1 Experiment configuration

We conduct a trace driven evaluation with data traces collected from USRP N210 software defined radios mainly because HOPE adopts Spinal codes which require excessive decoding overhead for the general software radio platforms.

**Experiment setup.** With the Spinal codes, a (delegated) source first divides each raw data packet into $k$-bit blocks. Each $k$-bit block is hashed to a $v$-bit spine value, which is then used as a seed to generate $c$-bit random numbers. Every "pass" is a concatenation of a set of $c$-bit random numbers. In particular, the $i$th "pass" for a data packet consists of the $i$th $c$-bit random number generated from each block of the packet. A receiver uses maximum likelihood decoder to recover the $k$-bit message from $c$-bit numbers. Our implementation of Spinal codes is based on the codes provided by the authors of [26] and we configure $k = 3$, $c = 4$, $v = 32$, and $B = 256$.

**Trace collection.** *Trace from USRP N210.* Each USRP node is configured to operate on OFDM with 600KHz frequency band in the 2.4GHz range. Each node uses a USRP RFX2400 daughterboard as the RF frontend with the transmission power of 50mW (i.e., 17dbm) attached to an omnidirectional antenna with 3dBi gain. We connect each node to a laptop to collect PHY symbols. The PHY of 802.11n in the single input single output mode is implemented on the GNU Radio platform. We collect the pairwise packet transmission traces of all 342 links among 19 positions in Fig. 8. The links include direct line-of-sight paths as well as those blocked by objects. We perform the trace collection with 19 rounds. In each round, one USRP sender is placed at one of the 19 locations, broadcasts the packets from a fixed 5MB data file, and circulates all data rates to transmit the same file. The rest 18 USRP receivers record all the received packets as the packet-level trace at the rest 18 locations simultaneously (no matter packets are successfully decoded or not). The packet-level trace directly records the fate of each packet from the 5 MB data file which is used to evaluate the packet-oriented protocols, SPR and ExOR. The USRP receivers further compare the received symbols with the transmitted ones for all received packets at all rates and compose a symbol-level trace of errors, which is used to test MIXIT and HOPE. By doing so, a 19-node network with all possible running traffic flows is constructed.

*Trace from Intel 5300 NICs.* USRP N210 can directly record the symbol-level traces, but mainly works in the narrow band channels due to the hardware constraints. We further adopt the channel state information (CSI) tool developed on Intel 5300 NICs [11] to collect transmission traces and test HOPE in the wideband channels with frequency selective fading. We deploy Intel 5300 nodes, operating in 20 MHz channel, at the same positions in Fig. 8 and perform the trace collection in the same way as we do with USRP N210s. Intel 5300 is only able to report packet-level traces. The dispersion of each symbol cannot be directly measured. To obtain the symbol-level traces for evaluating MIXIT and HOPE, when one node transmits, we let other nodes measure the subcarrier-level SNRs to the transmitter. For each transceiver pair, we use the measured SNRs to generate the symbol-level dispersions [11].

We perform trace-driven emulations with the collected 19-node network traces. We build a custom emulator to evaluate HOPE compared with benchmarks. The emulator implements 802.11-like PHY (e.g., convolutional coding and modulations) and MAC. We first experiment with the USRP trace to investigate the detailed performance of different approaches, and then use Intel 5300 trace to evaluate the wideband performance.

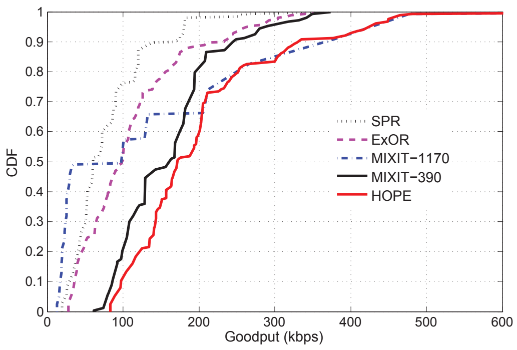

**Figure 9: Goodputs of various approaches.**

### 4.2 Results

**Goodput**. We evaluate the goodput that refers to the actual data size transmitted over unit time. We consider goodput over throughput as different protocols incur different amounts of protocol overhead, and goodput can better reflect the data transmission efficiency and delay, i.e., a higher goodput indicates better efficiency and a shorter delay [33]. On the other hand, to understand the protocol overhead, we further evaluate the detailed breakdown of throughput for each protocol in this section.

*Goodput comparison.* We evaluate the goodput performance of different protocols. We randomly select one node pair and measure its end-to-end goodput. For all benchmark protocols SPR, ExOR and MIXIT, we test all possible 802.11n single-stream data rates in the USRP 600KHz frequency band and select their best performance. Both SPR and ExOR achieve the maximum goodput with 390Kbps data rate. MIXIT achieves a better average goodput at 1.17Mbps, but has some unique advantage at 390Kbps data rate, so we plot both MIXIT-1170 and MIXIT-390. We set the data rate 1.17Mbps for HOPE, which yields a gradually decreasing effective data rate when more Spinal "passes" are transmitted. We repeat the experiments 120 times by randomly selecting 120 node pairs in total. Fig. 9 depicts the

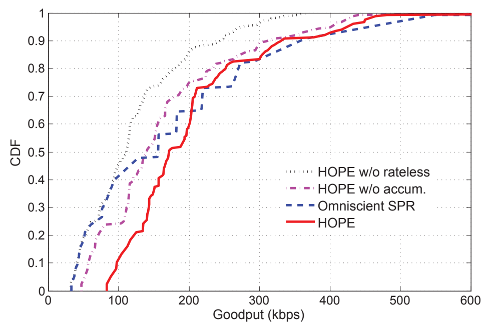

**Figure 10: Goodput gain breakdown of HOPE.**

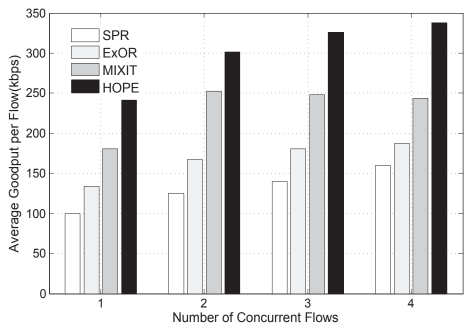

**Figure 12: Goodputs in a multi-flow mesh network.**

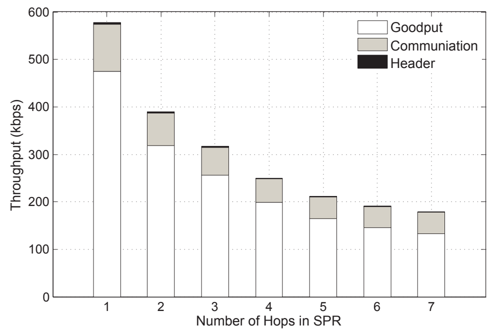

**Figure 11: Overhead breakdown of HOPE.**

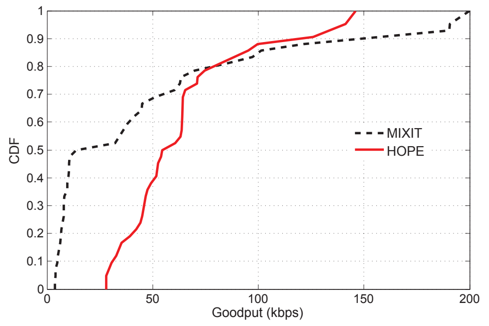

**Figure 13: Goodputs in a multi-flow ad-hoc network.**

CDF of the achieved goodput by different protocols. Generally HOPE achieves an average goodput gain of 2.4x over SPR, 1.7x over ExOR, 1.3x over MIXIT of both 1.17Mbps and 390Kbps, respectively. HOPE significantly outperforms SPR and ExOR primarily because HOPE allows each delegated source to adapt to all possible accumulators' channel conditions simultaneously and thus accumulators largely accelerate the packet decoding based on the cumulative information from multiple sources.

Along short paths or high quality links (mostly the right part of the figure), we see that both MIXIT-1170 and HOPE achieve high goodput because the channel condition supports their transmissions with full rates. Over long paths or weak links (mostly the left part of the figure), however, MIXIT-1170 suffers dramatic goodput drop. It is probably because MIXIT at 1.17Mbps data rate misses many low quality symbol transmissions and suffers high symbol error rate over weak links. The uncertain symbol errors may accumulate and propagate over long paths. If we set the data rate to 13Mbps on the other hand, MIXIT captures the symbol accumulation opportunities over weaker links, but cannot completely exploit good links with full rates. *Note that, as opportunistic routing protocols, e.g., MIXIT and ExOR, have no fixed receiver for each transmission. They cannot automatically adapt their own rates to different links of different qualities.*

*Gain analysis.* We test HOPE with three different settings to understand its goodput gain in detail. We first substitute the rateless codes of HOPE with a fixed data rate of 390Kbps, which degrades to capturing packet-level opportunities similar as ExOR (HOPE w/o rateless). We then enable rateless codes but disable the information accumulation of HOPE (HOPE w/o accum). Finally, we fully enable HOPE with both benefits. Fig. 10 depicts the performance of the three settings. With the rateless codes, HOPE harvests more than half of its full goodput gain. For good channel conditions (e.g., the goodput is higher than 240Kbps), the rateless codes play a critical role in acquiring the goodput. In Fig. 10, the goodput of HOPE w/o accum. is comparable to HOPE in this region. For the long paths or poor links, rateless codes also help in adapting different links and counteracting channel fluctuations. Enabling information accumulation allows HOPE to benefit more in the low goodput region. From Fig. 10 we see the main improvement of HOPE over HOPE w/o accum. when the goodput is lower than 240Kbps. This is because the cumulative information is more valuable for long paths and weaker links. In Fig. 10, we also examine omniscient SPR for comparison. Omniscient SPR always sets the optimal data rate for the packet transmission at each hop. HOPE achieves comparable goodput with omniscient SPR in the high goodput region due to the auto rate adaptation. For the long paths and weak links, HOPE outperforms omniscient SPR, as HOPE accumulates more overheard rateless units under such scenarios. We further find that HOPE can rapidly adapt to a better link when the channel condition varies while SPR lacks this flexibility.

**Overhead.** Fig. 11 examines the overhead of HOPE over paths of different hop counts. We separate the communication overhead and data header overhead from the goodput. The communication overhead consists of ACK transmissions, duplicate packet transmissions from different nodes, carrier sensing time, etc. The small and fixed-sized header in HOPE poses negligible overhead. The communication overhead is also small, $< 10\%$ of throughput, as HOPE does not have to spend time in arbitrating the next forwarder each time. According to the experiment results,

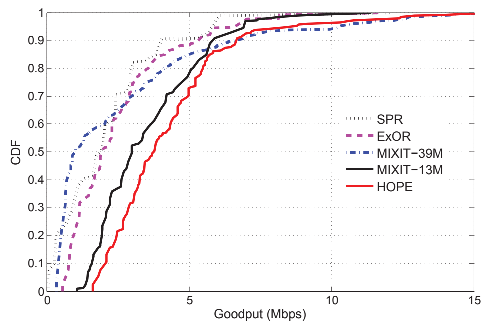

**Figure 14: Goodputs in wideband channels.**

we see that the fraction of overhead gradually increases as hop counts increase, which is mainly because the involved extra control overhead adds up every hop while the payload size stays constant at 1500bytes.

**Multiple flows**. We test the performance of SPR, ExOR, MIXIT and HOPE with multiple flows for mesh-like and ad-hoc-like networks. For MIXIT, we find that the 1.17Mbps rate outperforms the 390Kbps rate in this trial of experiments and thus only present its performance with the 1.17 Mbps rate. In Fig. 12, we examine the performance of each protocol in a mesh-like network with a gateway node (the highlighted node in Fig. 8) and other nodes incur the flow based traffic to the gateway. By setting different numbers of original sources, we can vary the number of flows in the network. Each original source incurs a traffic flow to the gateway for transmitting a 5MB file. Fig. 12 plots the total network goodput with varied numbers of flows. The network goodput increases towards the saturated capacity as the number of concurrent flows increases. HOPE, however, consistently outperforms benchmark protocols due to its best use of the wireless channel. According to statistics, HOPE achieves an average goodput gain of 2.3x over SPR, 1.7x over ExOR, 1.3x over MIXIT at 1.17Mbps.

We further evalaute HOPE and MIXIT (with highest goodput among benchmark schemes) in an ad-hoc network with intermittent traffics. All 19 nodes in Fig. 8 act as ad-hoc-like nodes and we configure 6 coexisting flows with different sender-destination pairs in the network but of randomly selected data volume ranging from 1.5-30K bytes. When one flow finishes, we immediately appoint a new flow between a random pair of nodes. We experiment 50 flows in total. Fig. 13 depicts the CDF of the goodput achieved by MIXIT and HOPE, respectively. In the high goodput region, MIXIT slightly performs better mainly because its congestion-aware scheme helps to approach the network capacity. The high control overhead and per-hop errors, however, limit the performance of MIXIT over longer paths. In the low goodput region HOPE achieves more efficient bandwidth utilization. Overall, HOPE achieves 1.4x average goodput gain.

**HOPE on wideband channels**. In this section, we experiment with the wideband traces collected from Intel 5300 NICs. We select one node pair and measure its end-to-end goodput. We repeat the experiments 120 times by randomly selecting 120 node pairs. To avoid long runs of consecutive errors in the coded bit sequences due to frequency selective fading, interleaving is implemented for each protocol. Fig. 14 shows the CDF of the achieved goodput of different protocols. Compared with the performance in narrowband channels (Fig. 9), the goodput gain achieved by HOPE is still substantial. In particular, HOPE achieves an average goodput gain of 2.0x over SPR, 1.6x over ExOR, 1.5x over MIXIT-39M, and 1.3x over MIXIT-13M. MIXIT-39M and MIXIT-13M represent MIXIT works at 39Mbps and 13Mbps and correspond to MIXIT-1170 and MIXIT-390 in Fig. 9, respectively. The gain achieved by HOPE over SPR and ExOR is preserved since it can still adapt to accumulators' channel conditions simultaneously even with frequency selective fading. Accumulators can thus leverage the cumulative information to accelerate packet decoding.

We also evaluate HOPE on low-power wireless networks. Due to the page limitation, the results can be found at [23].

## 5. RELATED WORKS

**Rate adaptation and rateless codes.** Rateless codes can achieve optimal rate adaptation [10, 26]. With rateless codes, a sender incrementally adapts to a suitable data rate by sending more coded data until the receiver decodes the data. Strider [10] presents a systematic design of automatic rate adaptation based on a layered rateless code with the linear combination of turbo codes. LT [24] and Raptor codes [30] can achieve Shannon capacity for binary erasure channels, but it remains unclear how close they are to the capacity over wireless channels. The authors in [8] further investigate how LT codes can be used to prolong the end-to-end communication range in low-power wireless networks. Departing from prior linear rateless codes, Spinal codes [26] use hash functions to produce infinite sequence of PHY symbols, achieving near optimal capacity over both erasure and wireless channels. Different from existing works, our design leverages rateless codes to adapt to joint information channels with hopless networking.

**Opportunistic routing.** ExOR [2] pioneers the opportunistic routing, which broadcasts packets without predetermined next hops and delays forwarding decisions to exploit channel diversity. MORE [3] obviates strict scheduling and enhances spatial reuse by leveraging network coding. Both ExOR and MORE can only make use of intact packets surviving over opportunistic links. SPaC [9] combines multiple corrupted packets in sensor networks to recover data. MIXIT [20] can forward clean symbols in corrupted packets and applies symbol-level network coding. The unified data rate of each transmission, however, limits the information utility of SPaC and MIXIT. In contrast, HOPE features rateless information dissemination and adapts to diverse channel conditions of all wireless links. MIXIT does study from a rateless perspective but relies on the end-to-end network coding to adapt to the network, which inevitably misses the full gain from internal channel diversities within the network. MIXIT requires an accurate network reliability prediction to avoid error propagation.

**Cooperative communication.** Our work is related to cooperative communication in which users coordinate to enhance communication quality [21, 28, 27, 32]. Laneman et. al [21] develop information-theoretic approaches to exploit wireless spatial diversity. Hunter *et al.* [17] propose to integrate cooperative communication with channel coding. Cover *et al.* [4] present the theoretical analysis on the capacity of relay channel. Draper *et al.* [7] design routing protocols for cooperative networks that perform mutual information accumulation. The major difference between [7] and HOPE is that the former study focuses on the optimization of the transmission order in theory, while our solution concentrates on addressing realistic challenges and designing a practical system incorporating the latest rateless codes, e.g., Spinal codes. In summary, existing researches mainly study cooperative communications from a theoretical perspective, e.g., gain analysis, schedule optimization, etc. In this paper, we propose a clean hopless networking abstraction from a system perspective, address a sequence of realistic design issues, and develop a practical solution using rateless codes to harvest cooperative diversity of autonomous wireless nodes which has never been done before.

## 6. CONCLUSION

This paper presents a novel hopless networking paradigm. The key idea is to break per-hop packet transmission to rateless information dissemination that adapts to diverse channel conditions of wireless links, taking the full advantage of wireless broadcast effect. We develop a practical protocol, HOPE, which instantiates the hopless networking. Experimental evaluation demonstrates that HOPE significantly improves network throughput over existing approaches. While the current HOPE design does not pursue the performance optimality, we point out several important future works, e.g., optimal rateless segmentation, better concurrency in multiple access, and flow-based optimization.

## 7. ACKNOWLEDGEMENTS

This work is supported by Singapore MOE AcRF Tier 2 Grant MOE2012-T2-1-070 and Singapore MOE AcRF Tier 1 grant MOE2013-T1-002-005. This work is also supported by the Singapore National Research Foundation under its Environment and Water Technologies Strategic Research Programme and administered by the Environment and Water Industry Programme Office (EWI) of the PUB on project 1002-IRIS-09, NSF CNS-1116970, and NSFC 61303233.

## 8. REFERENCES

[1] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *ACM MobiCom*, 2005.

[2] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *ACM SIGCOMM*, 2005.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.

[4] T. Cover and A. E. Gamal. Capacity theorems for the relay channel. *IEEE Trans. on Information Theory*, 1979.

[5] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom*, 2003.

[6] W. Dong, Y. Liu, Y. He, and T. Zhu. Measurement and analysis on the packet delivery performance in a large scale sensor network. In *IEEE INFOCOM*, 2013.

[7] S. C. Draper, L. Liu, A. F. Molisch, and J. S. Yedidia. Cooperative transmission for wireless networks using mutual-information accumulation. *IEEE Trans. on Information Theory*, 2011.

[8] W. Du, Z. Li, J. C. Liando, and M. Li. From rateless to distanceless: Enabling sparse sensor network deployment in large areas. In *ACM SenSys*, 2014.

[9] H. Dubois-Ferrière, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *ACM SenSys*, 2005.

[10] A. Gudipati and S. Katti. Strider: automatic rate adaptation and collision handling. In *ACM SIGCOMM*, 2011.

[11] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM SIGCOMM*, 2010.

[12] J. Han, C. Qian, X. Wang, D. Ma, J. Zhao, P. Zhang, W. Xi, and Z. Jiang. Twins: Device-free object tracking using passive tags. In *IEEE INFOCOM*, 2014.

[13] J. Han, C. Qian, P. Yang, D. Ma, Z. Jiang, W. Xi, and J. Zhao. Geneprint: Generic and accurate physical-layer identification for UHF RFID tags. *IEEE Trans. on Networking*, 2015.

[14] M. K. Han, A. Bhartia, L. Qiu, and E. Rozner. O3: optimized overlay-based opportunistic routing. In *ACM MobiHoc*, 2011.

[15] L. He, L. Fu, L. Zheng, Y. Gu, P. Cheng, J. Chen, and J. Pan. Esync: An energy synchronized charging protocol for rechargeable wireless sensor networks. In *ACM MobiHoc*, 2014.

[16] Q. Huang, K. Sun, X. Li, and D. Wu. Just fun: A joint fountain coding and network coding approach to loss-tolerant information spreading. In *ACM MobiHoc*, 2014.

[17] T. E. Hunter and A. Nosratinia. Diversity through coded cooperation. *IEEE Trans. on Wireless Commununication*, 2006.

[18] K. Jamieson and H. Balakrishnan. PPR: Partial packet recovery for wireless networks. In *ACM SIGCOMM*, 2007.

[19] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, 2000.

[20] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *ACM SIGCOMM*, 2008.

[21] J. N. Laneman, D. N. C. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Trans. on Information Theory*, 2002.

[22] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner. Predictable performance optimization for wireless networks. In *ACM SIGCOMM*, 2008.

[23] Z. Li, W. Du, Y. Zheng, M. Li, and D. Wu. From rateless to hopless. In *Technical Report*, 2015.

[24] M. Luby. Lt codes. In *Proceedings of IEEE FOCS*, 2002.

[25] Q. Ma, K. Liu, X. Xiao, Z. Cao, and Y. Liu. Link scanner: Faulty link detection for wireless sensor networks. In *IEEE INFOCOM*, 2013.

[26] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah. Spinal codes. In *ACM SIGCOMM*, 2012.

[27] A. Sendonaris, E. Erkip, and B. Aazhang. User cooperation diversity - part ii: Implementation aspects and performance analysis. *IEEE Trans. on Communications*, 2003.

[28] A. Sendonaris, E. Erkip, and B. Aazhang. User cooperation diversity: System description. *IEEE Trans. on Communications*, 2003.

[29] L. Shangguan, Z. Yang, L. Alex, and Y. Liu. Relative localization of rfid tags using spatial-temporal phase profiling. In *USENIX NSDI*, 2015.

[30] A. Shokrollahi. Raptor codes. *IEEE Trans. on Information Theory*, 2006.

[31] L. Wang, Y. He, Y. Liu, W. Liu, J. Wang, and N. Jing. It is not just a matter of time: oscillation-free emergency navigation with sensor networks. In *IEEE RTSS*, 2012.

[32] X. Wang, L. Fu, and C. Hu. Multicast performance with hierarchical cooperation. *IEEE/ACM Trans. on Networking*, 2012.

[33] X. Wang, W. Huang, S. Wang, J. Zhang, and C. Hu. Delay and capacity tradeoff analysis for motioncast. *IEEE/ACM Trans. on Networking*, 2011.

[34] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Liu, and M. Liu. Phaseu: Real-time los identification with wifi. In *IEEE INFOCOM*, 2015.