Drizzle: Cooperative Symbol-Level Network Coding in Multi-Channel Wireless Networks

Ronny Yongho Kim, *Member, IEEE,* Jin Jin, *Student Member, IEEE,* and Baochun Li, *Senior Member, IEEE*

***Abstract*—Errors are inherently present in unreliable wireless channels. The primary challenge in designing error control protocols in the MAC or physical layer is to effectively maximize achievable throughput in wireless networks even when unpredictable and time-varying errors exist. Network coding has been successfully applied to improve throughput in IEEE 802.11-based wireless networks with a shared broadcast channel. In state-of-the-art physical layer designs in multi-channel wireless networks (such as IEEE 802.16 WiMAX), however, the convenience of a shared wireless broadcast channel to perform opportunistic listening no longer exists, and Hybrid ARQ (HARQ) is the predominant error control protocol in the physical layer, rather than plain ARQ in IEEE 802.11 MAC. Would network coding be well employed in multi-channel wireless networks and able to bring further improvements over HARQ? This paper proposes *Drizzle*, a new solution to maximize throughput with the presence of errors, that takes advantage of network coding at the symbol level in multi-channel wireless networks. By operating at the symbol level and using soft decision values, we show that *Drizzle* is able to exploit both time and cooperative diversity in realistic multi-channel wireless networks, to adapt to time-varying and bursty channel errors, and to efficiently collect as many correct symbols as possible at the receiver.**

***Index Terms*—Network Coding, Soft Decision, Diversity, Modulation, Error Correction, Cooperative Transmission, WiMAX**

I. Introduction

Multi-channel wireless networks represent a direction that most future 4G state-of-the-art wireless communication standards evolve towards, including IEEE 802.16 WiMAX [1] and 3GPP Long Term Evolution (LTE) [2]. In both WiMAX and LTE, Orthogonal Frequency Division Multiple Access (OFDMA) is used at the physical layer. OFDMA uses a large number of orthogonal *subcarriers* to maximize spectral efficiency, and assigns different subsets to different users to achieve multiple access.

It is common knowledge that errors are inherently present in unreliable wireless channels. The important challenge in designing error control protocols in the MAC or physical layer is to effectively *maximize achievable throughput* in various transmission scenarios in wireless networks, even when unpredictable and time-varying errors exist.

Manuscript received June 08, 2009; revised September 24, 2009 and November 06, 2009.

R. Y. Kim is with LG Electronics, Anyang, Kyoungki 431-749 Korea (e-mail:ronnykim@lge.com).

J. Jin and B. Li are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: jinjin,bli@eecg.toronto.edu).

With respect to the objective of maximizing throughput, *network coding* has been originally proposed in information theory [3], [4], and has since emerged as one of the most promising information theoretic approaches to improve throughput. Network coding has been successfully applied in multi-hop wireless networks to opportunistically take advantage of multiple routes from the sender to the receiver in unicast flows [5], [6], and soft decision values from the physical layer are utilized to perform partial packet recovery when packets are broadcast in a shared IEEE 802.11-based wireless channel [7]. Unfortunately, in multi-channel wireless networks — such as IEEE 802.16 WiMAX with OFDMA at the physical layer — the convenience of a shared wireless broadcast channel to perform opportunistic listening no longer exists, and Hybrid Automatic Repeat reQuest (HARQ) is the predominant error control protocol at the physical layer [8], rather than plain Automatic Repeat reQuest (ARQ) in IEEE 802.11 MAC.

However, HARQ may not be able to effectively perform error control and under-utilize the scarce wireless bandwidth. As HARQ is designed for the point-to-point channel without flexibility, it might not utilize opportunities on concurrent multi-path transmissions, which are created by multi-channel wireless networks.

Would *network coding* still be helpful in multi-channel wireless networks? How do we design an efficient error control protocol for multi-channel wireless networks as they are the norm in the next-generation (4G) industry standards? In this paper, we present *Drizzle*, a new solution at the physical layer that uses network coding at the symbol level. *Drizzle* is carefully designed to fully embrace the characteristics of multi-channel wireless networks: rather than using network coding at the packet level (as in previous work in IEEE 802.11 networks), network coding in *Drizzle* is performed over *symbols* at the physical layer (a small sequence of bits in the physical layer).

When operating in the physical layer of multi-channel wireless networks such as WiMAX, *Drizzle* shows two salient advantages. *First*, the sender only needs to retransmit "dirty" symbols — the ones corrupted by channel errors after demodulation — rather than the entire packet. An illustrative example is shown in Fig. 1. The sender first divides each single packet into a number of (5 in the example) small *blocks*, each of which contains one or a small number of physical layer *symbols* used in modulation. All blocks are encoded using random network coding [9], [10], and the sender sends the packet by transmitting 5 of them ($A1, A2, \cdots, A5$) to the receiver. With random network coding, the sender is able to

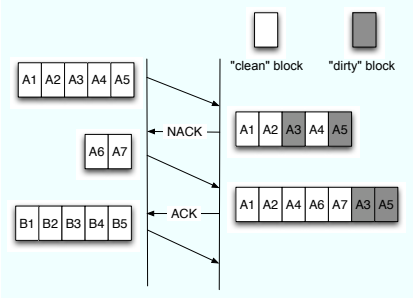


Fig. 1. In Drizzle, only "dirty" blocks are retransmitted to the receiver over a single wireless link with errors.

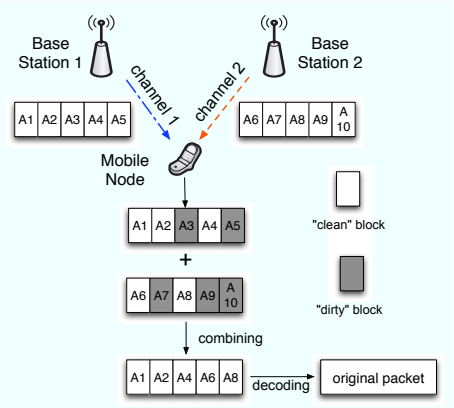


Fig. 2. Drizzle allows multiple senders to cooperatively send coded blocks to the same receiver in multi-path transmissions, such as the handover scenario in WiMAX.

generate a virtually unlimited number of coded blocks using different sets of coefficients, and any n (required number of blocks for decoding; n is 5 in the example) of these coded blocks can be used to perform decoding by inverting a matrix of coding coefficients. This is referred to as the rateless property, with which all the blocks within one packet are equally useful. Due to unreliable channels, the packet may be corrupted in the transmission. However, not all the bits within the packet share the same fate. Very often, only a small number of bits are in error; the rest are correct. In the example, block $A3$ and $A5$ are in error, while $A1$, $A2$ and $A4$ are "clean." Under this situation, the sender just needs to send two more coded blocks ($A6$ and $A7$) to the receiver, which can then be used towards correct decoding of the packet on the receiver, with a total of 5 "clean" blocks received.

Clearly, as the size of a *block* is sufficiently small, error control in *Drizzle* can be performed in fine granularity, which can be more efficient in terms of resource utilization than traditional packet-level error control protocols and blind-push based end-to-end error correction in [7]. In addition, thanks to the rateless property of random network coding, the receiver does not have to specify which blocks have errors in the packet, and only needs to ask for an additional number of blocks. Should a particular coded block be lost, subsequent correctly received ones are equally innovative and useful to recover the original packet. As such, Drizzle is resilient to time-varying and bursty channel errors, by dynamically adapting to fluctuating channel conditions in realistic networks such as WiMAX, especially when mobility is present.

Second, *Drizzle* works best in multi-hop multi-channel wireless networks, such as handover and multi-hop modes in WiMAX. In such networks, a mobile node is able to establish connections with two or more upstream nodes through different sub-channels (different subsets of orthogonal subcarriers in OFDMA). Cooperatively, they can use different sets of coefficients to generate coded blocks for the same transmitted packet. As an example shown in Fig. 2, base station 1 generates coded blocks $A1, \cdots, A5$, and base station 2 produces $A6, \cdots, A10$ similarly. The mobile node is able to "collect" coded blocks from both connections simultaneously without interference and try to decode the packet by combining "clean" coded blocks. Although there are "dirty" blocks in each reception — $A3$ and $A5$ from base station 1, $A7, A9$ and $A10$ from base station 2 — the mobile node is still able to reconstruct the packet from errors by collecting sufficient number of "clean" coded blocks ($A1, A2, A4, A6$ and $A8$). Again, due to the rateless property of random network coding, it is not required to use sophisticated channel estimation and allocation mechanisms to dictate where these blocks should come from. In Drizzle, "clean" coded blocks from any of the senders are equally useful. With Drizzle, the mobile node is able to enjoy concurrent multi-path transmissions by dynamically "collecting" fine "rain drops," which will improve the throughput performance significantly.

How does *Drizzle* distinguish "clean" symbols from "dirty" ones? *Drizzle* takes advantage of soft decision values provided by physical layer demodulation on each bit received, and estimates the correctness of a symbol after demodulation, considering the adaptive modulation schemes being used and channel conditions. Throughout the remainder of this paper, we use the IEEE 802.16 WiMAX family of standards as a representative of physical layer design in multi-channel wireless networks. We seek to demonstrate the advantages of *Drizzle* in WiMAX, and we believe these advantages will hold when *Drizzle* is applied to other multi-channel wireless networks based on OFDMA and HARQ.

The salient highlight of our work is a novel framework for error control in multi-channel wireless networks that exploits all the potential benefits above. To achieve such an objective, there are a number of challenges:

- ▷ How is Drizzle integrated with the existing techniques adopted at the physical layer of multi-channel wireless networks?
- ▷ How does Drizzle accurately estimate the correctness of each received coded block using soft decision values conveyed from the physical layer?
- ▷ How does Drizzle minimize the overhead generated in the transmission process?

Our responses to these challenges constitute the flow of presentation in this paper. In Sec. II, we review related work on the use of network coding in wireless networks. From Sec. III to Sec. V, we present the design of *Drizzle*. We provide an analytical comparison between Drizzle and HARQ in Sec. VI. We evaluate the performance of Drizzle in WiMAX networks in Sec. VII, and show that *Drizzle* offers important advantages as compared to HARQ and previous work in the literature, which use retransmission based error recovery. Finally, we conclude the paper in Sec. VIII.

II. Related Work

In the WiMAX physical layer, Hybrid Automatic Retransmission reQuest (HARQ) is adopted as an error control protocol by combining ARQ and Forward Error Correction (FEC) [8]. In Type-II and Type-III HARQ, its performance can be further improved by packet soft combining [11], [12]. Its performance, especially in the context of WiMAX, has been thoroughly investigated in an information-theoretic fashion [13], [14]. However, the built-in reliability in HARQ sacrifices some degree of *resilience* to time-varying channel conditions [15]. In addition, HARQ does not exploit the cooperative diversity in multi-path transmissions, as it is designed for a point-to-point channel. Drizzle intends to serve as a replacement of HARQ in the WiMAX physical layer. In this paper, we evaluate our protocol against HARQ, which is well tuned and has been offering satisfactory performance in WiMAX.

A parallel multi-path transmission strategy over multiple network interfaces, referred to as MuniSocket, is studied in [16]. MuniSocket is a middleware solution to provide efficient packet transmission over heterogeneous networks. MuniSocket divides a packet into multiple fragments and transmits them using multiple TCP connections over multiple network interfaces in parallel. It is shown that MuniSocket is able to improve the throughput by taking advantage of parallel transmission. However, MuniSocket is not specially designed for wireless networks, especially multi-channel wireless communication systems. Different from Drizzle that works in the physical layer, MuniSocket works in the transport layer. Due to error-prone wireless channels, packet errors will frequently trigger TCP congestion control and retransmissions in MuniSocket, which would degrade the throughput dramatically.

In the context of 802.11-based wireless networks with a single, shared wireless broadcast channel, a partial packet recovery algorithm proposed by Jamieson *et al.* [17] has been proposed to revise the traditional ARQ. Rather than retransmitting the entire packet, the erroneous portions of the packet would be retransmitted. In some sense, this is akin to the general idea of HARQ in WiMAX, except that the feedback message has to explicitly describe the positions of error bits in the packet, which would likely incur significant overhead. Further, it is not designed to support cooperative transmissions in a typical multi-path transmission scenario.

Woo *et al.* proposed a cooperative packet recovery algorithm in 802.11-based networks, referred to as SOFT [18]. SOFT works by combining confidence values across multiple faulty receptions to recover a clean packet. It is shown that SOFT is able to significantly improve the data delivery rate in 802.11-based networks, in static wireless environments. However, we believe that realistic channel conditions are time-varying and bursty in multi-channel wireless networks, such as WiMAX networks. The performance of SOFT under such conditions is unclear. Different from SOFT, Drizzle uses network coding at the physical layer that provides *resilience* to errors. It helps to better adapt to time-varying wireless channels in WiMAX, and is designed for cooperative transmission by multiple senders.

Network coding has been successfully applied in wireless networks to opportunistically take advantage of multiple routes from the sender to the receiver in unicast flows [5], [6]. Authors in [19], [20] proposed to use network coding in the physical layer. Similar to XOR of two packets in bits, these works perform XOR in the physical wireless radio signal level. Amplitudes and phases of wireless signals can be combined. The receiver is able to decode the desired physical wireless signal if it knows the other combined signals. In [15], MAC layer Random Network Coding (MRNC) has been introduced to avoid the overhead problems incurred by HARQ. Alamdar *et al.* proposed pre-coded transmission scheme using random network coding rather than frequency diversity, achieving significant performance improvement [21]. Stability analysis of random network coding across multicast sessions has been well studied in [22]. These works take advantage of the rateless property of random network coding: all data blocks are encoded as the random linear combination of the original packets and all independent coded blocks are equally useful and innovative [10], [23].

S. Katti *et al.* proposed MIXIT [7], a protocol for cooperative packet recovery by performing opportunistic routing on groups of correctly received symbols in a packet. MIXIT takes advantage of the broadcast nature of 802.11-based wireless networks and perform random network coding across correct symbols in different packets. MIXIT provides end-to-end error recovery by employing Maximum Rank Distance (MRD) codes [24] for push based blind redundancy transmission. However, it heavily relies on opportunistic listening and routing properties in multi-hop 802.11 networks, and can not be effectively applied to multi-channel wireless networks, such as WiMAX. Moreover, due to the bounded MRD code rates, it generates a large amount of overhead and is not able to provide flexibility on feedback based on-demand retransmission.

Our work differs from MIXIT in a number of aspects. *First*, we jointly employ random network coding and soft decision values. With such a proposed mechanism, random network coding is performed across the symbols within one packet rather than over different packets in MIXIT. Thus, the fine error control granularity of soft decision values and the favorable rateless property of random network coding can be both fully exploited, and are potentially helpful to improve the performance significantly in multi-channel wireless networks. *Second*, *Drizzle* can be implemented with low communication costs. As compared to the traditional network coding scheme, *Drizzle* employs new techniques including inner-packet coding, pre-generated coefficient matrix, and dynamic retransmission, with which signaling overhead and unnecessary redundant data transmission can be largely mitigated. According to our estimate, *Drizzle* is akin to a "free lunch" with respect to the computation and communication overhead with currently available technologies. *Third*, *Drizzle* is tightly designed for practical multi-channel wireless networks (*e.g.*, OFDMA based WiMAX), while providing flexibility to be applicable to other types of wireless networks. With the design of *Drizzle*, we seek to provide the answer to the question on whether network coding would provide additional improvements in *multi-channel* wireless networks, which is particularly interesting since HARQ is readily used in these physical layer protocols, with exceptional performance of

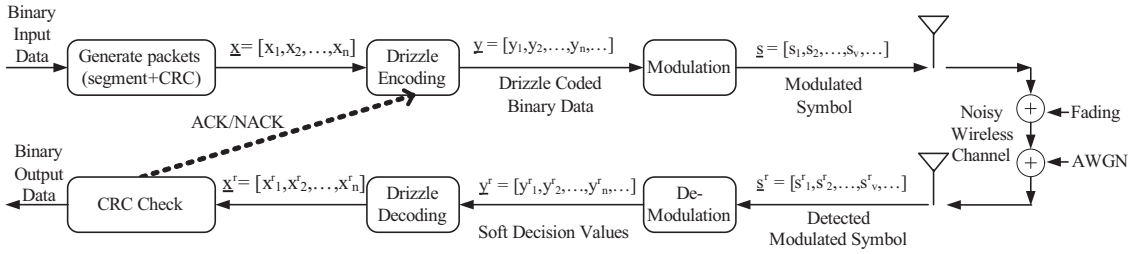


Fig. 3. A simplified block diagram showing the design of Drizzle.

using the available wireless bandwidth.

III. The Design of Drizzle

Drizzle is designed specifically to explore the benefits of using network coding at the symbol level in the physical layer of multi-channel wireless networks, with IEEE 802.16 WiMAX [1] as a representative example in this paper. The symbol level design of Drizzle allows for flexible and efficient operations, as compared to the rigid design of previously proposed physical layer network coding in [19], [20]. A *symbol* described in this paper refers to a unit of data that is defined by the modulation scheme in the physical layer. For example, one symbol represents two bits if Quadrature Phase Shift Keying (QPSK) is used, and four bits if 16 Quadrature Amplitude Modulation (16QAM) is used.

A. Basic Operations

In order to provide a good understanding of Drizzle, a simplified block diagram is shown in Fig. 3. The transmitter divides the input bit stream into segments and adds cyclic redundancy check (CRC), which is used for error detection at the receiver. A CRC appended segment is referred to as a *packet*. Each packet is then divided into *blocks* with fixed size ($\underline{\mathbf{x}} = [x_1, x_2, ..., x_n]$), each of which containing a certain number of physical layer symbols. We can easily compute the number of blocks in one packet if the packet size is predetermined, and we denote this quantity as the *batch size* in network coding. Unlike MIXIT [7], Drizzle performs random network coding upon blocks within the same packet. Let n be the batch size, and let x_i $(i = 1, 2, \cdots, n)$ be the blocks in the packet, c_{ji} $(i = 1, 2, \cdots, n)$ be the set of random coefficients generated in a given Galois field, the size of which is determined by the number of bits in a block (*e.g.*, for a block with 8 bits, GF(2^8) would be used). A coded block y_j can then be produced as $y_j = \sum_{i=1}^{n} c_{ji} \cdot x_i$. Each generated coded block can be mapped to one or several modulation symbols. The required number of symbols for one coded block depends on the size of the coded block and the selected modulation scheme. For example, a coded block with a block size of 8 bits is mapped to four symbols for QPSK and two symbols for 16QAM. The encoder is able to generate a virtually unlimited number of coded blocks y_j $(j = 1, 2, \cdots)$ using different sets of coefficients, and any n of these coded blocks can be used to decode by inverting a matrix of coding coefficients. This is usually referred to as the *rateless* property.

Demodulation in the physical layer on the receiver makes its best decision on the received signals. Due to noise and channel fading, the demodulator may make incorrect decisions, leading to errors. The Drizzle decoder tries to decode the received coded blocks using "hints" from demodulation, which are referred to as *soft decision values*. Soft decision values are estimations of code bit log likelihood ratios (LLRs) [25]. In the case of perfect channel knowledge, the estimation of code bit LLR under 2^F-QAM can be obtained by the following equation [25]:

$$\Lambda(b_f) = \ln \sum_{s^+ \in \{s:b_f=+1\}} \exp\left(-\frac{|y_s - \alpha s^+|^2}{\sigma^2}\right) - \ln \sum_{s^- \in \{s:b_f=-1\}} \exp\left(-\frac{|y_s - \alpha s^-|^2}{\sigma^2}\right) \quad (1)$$

where f is the bit order of used 2^F-QAM symbol; y_s is the received QAM symbol; α is the channel gain; s ($s \in \{s_1, s_2, \cdots, s_{2^F}\}$, $s = b_1 b_2 \cdots b_F$) is the transmitted QAM symbol; σ^2 is the variance of noise, which is a complex Gaussian random variable with zero mean. Fig. 4 shows 16-QAM constellation with Gray coding and an example of the detected symbol of y_s ($s = b_1 b_2 b_3 b_4 = 1001$). The first bit decides whether the detected symbol is placed in the first quadrant or fourth quadrant. For the first bit of a symbol ($b_1 = 1$), Eq. (1) will return a positive value because the detected symbol is placed in the fourth quadrant. The fourth quadrant is further divided into two spaces in the x-axis. If the detected symbol is placed in the left half of the divided space, LLR is a negative value, otherwise, LLR is a positive value. Through a similar approach, the third and fourth bits can be decided. From Eq. (1), it is clear that the shorter the Euclidean distance between the detected symbol and its closest constellation points is, the larger LLR value obtained.

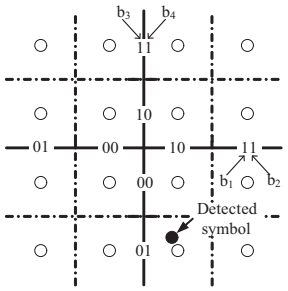


Fig. 4. 16-QAM (2^4-QAM) constellation with Gray coding and an example of detected symbol, 1001.

Essentially, soft decision values represent how much confidence the demodulator has in making the 0-1 decision on each bit. In Drizzle, an adaptive error detection algorithm is used to estimate the correctness of received blocks using these soft decision values from the demodulation process. We will

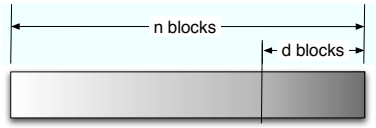


Fig. 5. A packet with "clean" and "dirty" blocks in Drizzle. The darkness indicates how much the block is polluted.

present a detailed discussion on the use of soft decision values in Sec. IV. With such estimates, Drizzle gives priorities to blocks with high confidence that they are correct, or "clean." It is important to have a sufficient number of "clean" blocks (with high probability), as many as the batch size n, before decoding begins, as "dirty" blocks will lead to decoding failures, which can be verified by checking CRC. When an error occurs, the receiver asks the sender(s) to retransmit additional coded blocks, until the entire packet is correctly decoded, or a maximum number of retransmissions is reached. When the packet can not be correctly recovered without a sufficient number of "clean" blocks until a maximum number of retransmission is reached, the packet is discarded at the physical layer. This strategy is employed for HARQ in various air interface standards including IEEE 802.16 WiMAX and 3GPP LTE.

B. Adaptive Retransmission

One of key designs in Drizzle is *adaptive retransmission*. Each received packets is inspected, and a confidence level of each block i (referred to as χ_i) in the packet is derived using soft decision values from the demodulation process as follows:

$$\chi_i = \frac{\sum_{t=1}^{T}\sum_{f=1}^{F}\left|\Lambda^{'}(b_{tf})\right|}{m} \tag{2}$$

where m is the total number of bits in one single block and $\Lambda^{'}(b_{tf})$ is normalized soft decision value of fth bit in tth symbol in the block. The normalized soft decision values is specially introduced in Drizzle to more accurately evaluate the symbol correctness, which will be discussed further in Sec. IV. There are F bits in one symbol and T symbols in one block. Clearly, $T \times F = m$. Essentially, Eq. (2) shows that the confidence level is calculated as the average value of normalized soft decision values of all bits in the block.

The blocks with lower confidence levels have lower priorities in the decoding process. An example is shown in Fig. 5, in which a darker block indicates a lower confidence level. The receiver constructs a set of blocks to decode, which always includes top n (batch size) blocks with the highest confidence levels, i.e., top n blocks with highest χ_i. If the decoding process fails, the receiver tries to exclude blocks with confidence levels below a certain threshold, marked as "dirty" blocks. In the example of Fig. 5, there are n received coded blocks in total, where d of them are classified as "dirty."

If decoding fails for the initial transmission, the receiver computes the number of "dirty" blocks (in this case d), and requests the sender to transmit additional coded blocks via NACK. The number of "dirty" blocks is determined using the level-threshold, which will be discussed and elaborated in Sec. IV. If the confidence level values of a block (χ_i) is

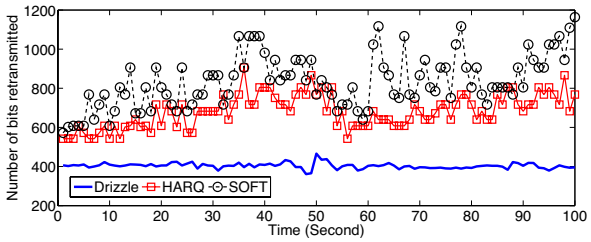


Fig. 6. The average number of bits retransmitted in a single-link transmission, when Drizzle is compared with HARQ and SOFT (Woo *et al.* [18]). Simulations are performed with the environment and settings provided in Sec. VII.

below the level-threshold (χ_{th}), i.e., $\chi_i \leq \chi_{th}$, the block is marked as "dirty." After receiving d additional blocks from the sender, the receiver has so far received $n+d$ blocks, and again tries to decode the packet with the n (out of $n+d$) blocks with the highest confidence levels. This process is referred to as *adaptive retransmission*, since the sender is only called upon to retransmit a sufficient number of additional blocks for the receiver to decode, and if blocks are sufficiently small, the available wireless bandwidth is effectively used, as in the analogy where fine "rain drops" fill up a "bucket."

We note that the receiver needs to identify the set of coded blocks from different packets, since decoding can only be performed with coded blocks from the same packet. For this purpose, a sequence number should be used to uniquely identify a packet. This is usually not an issue in physical layer designs. For example, WiMAX employs control channels — called Down Link (DL) Media Access Protocol (MAP) and Up Link (UL) MAP messages [1], which are allocated periodically at the beginning of the frame. Drizzle may use these control channels to deliver the sequence numbers of packets (similar to HARQ).

Adaptive retransmission is always performed as Drizzle is employed. How effective is this design to saturate available wireless bandwidth from the physical layer (after demodulation)? In one of our simulations, we have used a Rayleigh fading channel to simulate time-varying channel conditions between a sender and a receiver. With this channel, we simulated Drizzle, HARQ and SOFT [18] with WiMAX physical layer characteristics. Fig. 6 shows the number of bits retransmitted for correctly recovering the error packet over a period of time (100 seconds). We are able to observe that Drizzle consistently uses a significantly smaller number of bits in its retransmissions (on average 403 bits for Drizzle, 632 bits for HARQ, 835 bits for SOFT) which result in substantial throughput enhancement, and outperforms both HARQ and SOFT by 36% and 52%, respectively. The intuition is that, Drizzle allows the sender to retransmit a barely sufficient number of symbols, rather than blindly retransmitting the redundancy.

In terms of delay performance, Drizzle can achieve a shorter packet delivery time than HARQ and SOFT, since it transmits significantly smaller number of bits in its retransmissions, with shorter transmission delays. However, in the WiMAX Time Division Multiplexing (TDD) mode, the only deployed mode at the time of writing this paper, since the receiver has to wait for an uplink transmission opportunity to send ACK/NACK feedback to the transmitter, the gain on the transmission delay time reduction is negligible. Therefore, we focus on the

evaluation of throughput performance in this paper.

We will further evaluate the performance of adaptive retransmission in Drizzle in Sec. VII with more details.

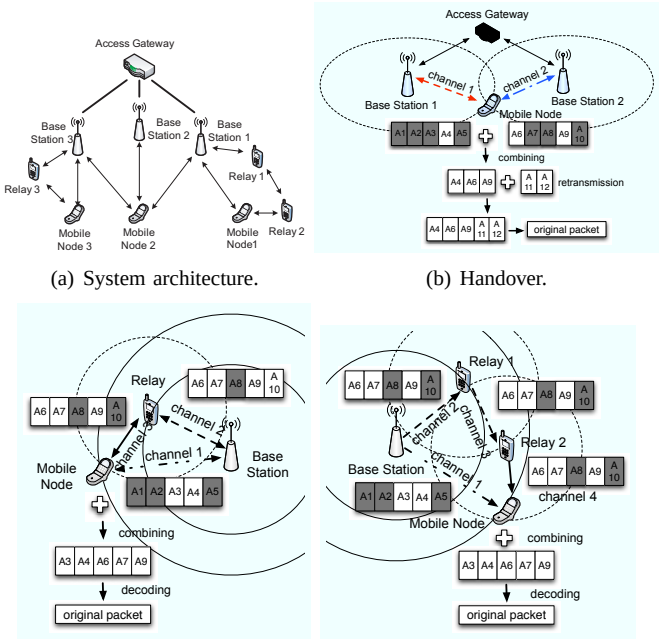
C. Cooperative Transmission

Cooperative transmission is specially designed for Drizzle to realize the potential benefits in multi-path transmission. Drizzle employs a typical wireless network architecture, as shown in Fig. 7(a), in order to provide an efficient and cost-effective cooperative transmission mechanism. In multi-channel wireless networks, such as WiMAX, a mobile node may frequently move across the boundary, and migrate from the air interface of one upstream node to that provided by another. In the overlapping region, a mobile node is able to connect to multiple upstream nodes. This is usually referred to as the *handover* scenario.

An illustrative example of handover is shown in Fig. 7(b), where a mobile node is in the handover region and connected to two upstream nodes (base station 1 and 2). By assigning separate sub-channels on each connection (channel 1 and 2), the receiver could communicate with all the upstream nodes concurrently with little interference, as sub-channels are orthogonal to each other by using OFDMA. As an opportunity for multi-path transmission is created in such scenarios, multiple senders are able to cooperatively transmit coded blocks to a receiver.

We use the example shown in Fig. 7(b) to show how cooperative transmission performs. In the figure, the access gateway mediates two base stations as a cross router and serves as the sender. It generates different coded blocks and send them to the base stations — $A1, \cdots, A5$ to base station 1, $A6, \cdots, A10$ to base station 2 (the batch size is set to be 5 in the example). The base stations forward the coded blocks and the mobile node receives them concurrently via different channels. The mobile node is able to "collect" different coded blocks from all the connections simultaneously. All the correctly received coded blocks are equally useful, due to the rateless property of random linear codes. However, as channels are not reliable, the mobile node only receives 3 ($A4, A6$ and $A7$) "clean" blocks (the dark blocks are "dirty" blocks, and the white blocks are "clean" ones), and thus fails to decode. It asks for retransmission, and the sender pushes more redundancy ($A11$ and $A12$). By correctly receiving the required number of correct blocks, the mobile node is able to successfully decode and recover the original packet. Such cooperative transmission could also be performed at the uplink, where the mobile node is responsible to generate distinct coded blocks, and the access gateway performs the decoding process.

Maximum-performance cooperative transmission can be employed for the downlink, as power is not a problem for relays and BSs. On the other hand, power-efficient cooperative transmission should be employed for the uplink due to mobile node's limited battery power. In the downlink, each upstream node uses different coefficient matrix to generate coded blocks and transmits generated coded blocks with different radio resources. However, in the uplink, the mobile node multicasts coded blocks to different upstream nodes. Due to different



(a) System architecture. (b) Handover.

(c) Two hops. (d) Three hops.

Fig. 7. Cooperative transmission of coded blocks is possible when the opportunity of multi-path transmission arises in both handover and multi-hop modes of multi-channel wireless networks.

position and fading environment of different upstream nodes, the received data experience different fading.

In addition, cooperative transmission also works well in the *multi-hop* mode of multi-channel wireless networks when relays are enabled. When the mobile node moves into an overlapped region covered by both an upstream node and a relay, the sender, receiver and relay are connected to one another via different sub-channels, where transmissions suffer little from interference. Having more than a single wireless hop, the *multi-hop* mode also creates an opportunity for multi-path transmission. The intuition is shown in Fig. 7(c), where the base station serves as the sender by issuing $A1, \cdots, A5$ directly to the mobile node and $A6, \cdots, A10$ to the relay, which will forward the data to the mobile node. By collecting the data from both paths concurrently, the mobile node tries to recover the original packet by decoding the "clean" coded blocks it receives. Moreover, cooperative transmission can be easily applied to more complicated topologies as shown in Fig. 7(d). Drizzle aims to take advantage of both random network coding and the convenience of multiple channels, and exploit the benefits of cooperation in multi-path transmission, that leads to the efficient use of available channel bandwidth.

As in the handover area, one mobile station can have multiple connections with upstream nodes. Concurrent multi-path transmission can be applied and will help to increase the throughput performance, as there are always data in the backlog for transmission. Thus, one issue we may be concerned with is *the amount of data each sender pushes to the receiver*. We observe that different channels experience different qualities, and sometimes the difference is quite significant. Drizzle takes advantage of such channel diversity and efficiently transmits the data. At the same time, Drizzle should provide fairness in resource allocation among users. Asking only one sender with the best channel quality to transmit all

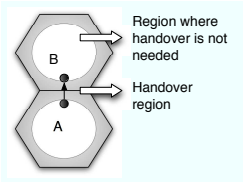


Fig. 8. Simulating a WiMAX handover event when an MS moves across the handover region with a constant speed.

the required data may cause the problem of starvation of some other users due to limited resources.

With the multi-path transmission, we propose the following scheme to determine the amount of data each sender transmits. Drizzle adopts a modified proportional rate constraint algorithm [26], [27], [28]. Assume that the channel Signal to Noise Ratio (SNR) is perfectly estimated. Denote the channel qualities of q channels, serving the same receiver, as $SNR_1, \cdots, SNR_q$. The number of coded blocks that need to be transmitted is denoted as N_R. The total number of coded blocks transmitted by each sender is denoted by N_i ($i \in \{1, \cdots, q\}$), *i.e.*, $N_i = \lceil N_i' \rceil$ where N_i' is computed as follows:

$$\frac{N_1'}{\beta_1} = \frac{N_2'}{\beta_2} = \cdots = \frac{N_q'}{\beta_q}$$

where

$$N_R = \sum_{i=1}^{q} N_i'$$

$$\beta_k = \frac{SNR_k}{\sum_{m=1}^{q} SNR_m}$$

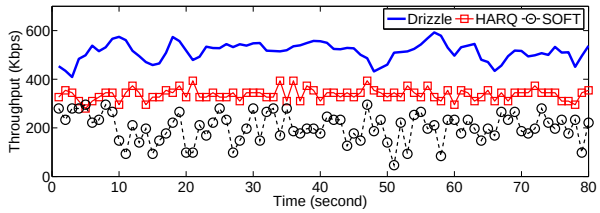
$$\sum_{j=1}^{q} \beta_j = 1.$$

We round up N_i' into integer values if N_i' are fractional numbers. Though this algorithm might require transmitting a few more blocks, Drizzle uses substantially less wireless resources for error correction as compared to HARQ and SOFT. The transmission of each sender is coordinated by either the access gateway or the base station, depending on the transmission scenarios (single link transmission, handover, or multi-hop transmission).

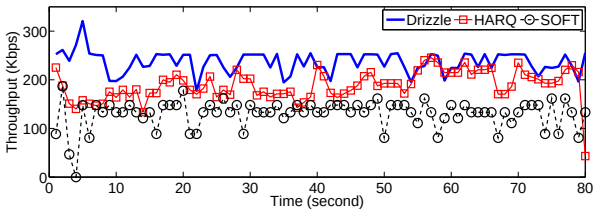
To show the benefits of Drizzle in cooperative transmission, we evaluate Drizzle using simulations in the handover scenario of WiMAX, against both HARQ and SOFT [18]. A mobile node moves across the handover region, from point *A* to point *B* in Fig. 8, with a constant speed. We measured the throughput on both downlink and uplink in our simulations. The results are shown in Fig. 9, where it is evident that Drizzle outperforms both HARQ and SOFT. On the downlink, Drizzle has an average throughput gain over HARQ and SOFT of 52% and 154%, respectively. On the uplink, this margin of improvement could reach 26% and 82%. Such substantial improvements coincide with our intuition that cooperative transmission in Drizzle naturally takes advantage of cooperative diversity in multi-path transmissions. We will examine the benefits of Drizzle in more practical multi-path transmission scenarios in Sec. VII.

IV. Impact of Soft Decision Values

Soft decision values conveyed from the demodulation process in the physical layer are used in Drizzle to detect errors in coded blocks. As described in Sec. III, we use Eq. (2) to obtain the confidence level of each coded block on the receiver. Is the confidence level able to fully capture the correctness of the block? Why do we use normalized soft decision values to calculate the confidence levels? How to take full advantage of these soft decision values in Drizzle? In this section, we present the design and the use of soft values, serving as important cornerstones in Drizzle.



(a) Downlink throughput



(b) Uplink throughput

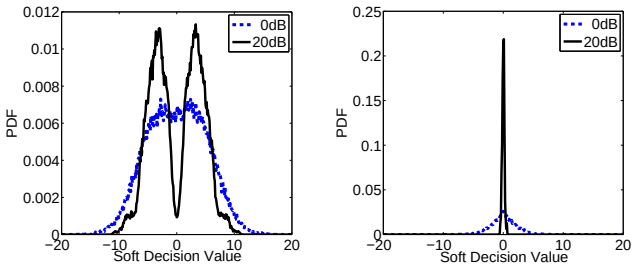
Fig. 9. The throughput performance of Drizzle over time (80 seconds) in the handover scenario, as a mobile station is moving around in the handover region randomly. Simulations are performed with the settings provided in Sec. VII.

A. Are Soft Decision Values Accurate?

Current modulation schemes in the physical layer compute the soft decision values (SVs) of all bits, which show the confidence of demodulation in order to make 0-1 decisions. A bit with a negative SV is translated into 1, whereas a bit with a positive SV is translated into 0. A larger absolute value in the SVs indicates a higher level of confidence on the decision being made.

Unfortunately, the distribution of SVs varies depending on modulation schemes and channel conditions. To further understand this point, we carried out a simulation. In the simulation, a packet of 25K bytes is transmitted over Rayleigh fading channels with a 30 km/h moving speed, and under different channel conditions. SV distributions of all received bits and error bits are shown in Fig. 10. As the figure shows, the SV distribution is different as the channel quality changes. For example, if we receive a bit with SV of -5 under the SNR of 0dB, there is still some probability that this bit is erroneous according to the SV distribution for error bits. However, the bit with SV of -5 is 100% “clean” under the SNR of 20dB. Also, different modulation schemes generate different SV distributions. Thus, it is not accurate to quantitatively measure the confidence levels without considering the impact of channel conditions and modulation schemes.

Intuitively, normalizing the soft decision values by considering different signal qualities and modulation schemes is a good solution to this problem. In Drizzle, soft decision values are normalized with the following formula:



(a) SV distribution of all bits (b) SV distribution of bad bits

Fig. 10. The distribution of soft decision values under BPSK modulation, which is obtained by transmitting 200,000 bits over Rayleigh fading channel with a speed of 30km/h.

$$NSV(s, SNR, M) = \frac{s}{|s|} \int_{-|s|}^{|s|} d(s, SNR, M) \mathrm{d}s \quad (3)$$

where NSV denotes the normalized SV, $d(s, SNR, M)$ denotes the probability density function (PDF) for SV under a certain SNR and modulation (M), and s denotes the SV variable. For example, $NSV(-10, 0dB, BPSK) = -96.3\%$. The normalized SVs are essentially a cumulative fraction of the absolute SV, since the SV distribution is symmetric with respect to 0. After normalization, the range of SV resides in $[-1, 1]$. It is straightforward that larger absolute values of the normalized SVs indicate higher confidence levels on the correctness of demodulation.

SV distributions under different channel qualities and modulation schemes are obtained from a large number of simulations. Normalized SVs are able to reflect relative characteristics of SVs, as they are tightly integrated with fluctuating channels and the adaptive modulation scheme adopted in the physical layer of multi-channel wireless networks. SVs in the remainder of the paper are normalized values if not noted otherwise.

B. How to Use SV for Error Detection?

In Drizzle, soft decision values have two main functions. *First*, they are used to construct the set of coded blocks used for decoding. To determine the confidence level of a coded block, Drizzle uses the absolute value of the normalized SV of each bit, and then computes the average of all bits in the coded block, as we already show in Eq. (2). A smaller average represents a lower confidence level that the block is correct, whereas a larger average shows a higher confidence level. As we have shown, blocks with higher confidence levels will be given higher priorities to be included in the set of blocks for decoding.

Unfortunately, confidence levels of coded blocks directly computed from the average may not be sufficiently accurate, since very often there exists a large variance on the absolute SVs of bits within one block (a block contains a small number of bits). A few bits with low absolute SVs may not effectively reduce the confidence level of the entire block, provided that there are much higher absolute SVs on some of the bits in the block. As such, it is necessary to penalize the blocks with one or a few "dirty" bits with low absolute SVs. The existence of even one block that is not correctly received will contaminate the entire decoding process.

In Drizzle, we check the SVs for all bits in a block. If any of the bits has an absolute SV that is below a certain threshold, we will set the confidence level of the entire block to the absolute SV with the lowest value. In this way, priorities of blocks with just a few error bits will be reduced, which provides a more accurate measure with respect to the confidence level of each coded block.

It is noted that this threshold (referred to as *SV-threshold*) should be carefully selected. We have studied the impact of the selection of the SV-threshold using numerical analysis and simulations.

For a given block error rate, P_B, packet delivery rate (P_S) can be expressed as following:

$$P_S = \sum_{j=n}^{K} \binom{K}{j} (1 - P_B)^j P_B^{K-j} \quad (4)$$

where j is the received number of blocks without error, K is the total number of transmitted blocks, and n is the batch size.

Let us denote the probability of bit error as P_e. Then, the probability of marking a correctly received bit as an erroneous bit is denoted as P_{CE} and the probability of marking an erroneous bit to be a correct bit as P_{EC}. The block error rate (P'_B) considering the mis-detection can be stated as:

$$P'_B = 1 - (1 - (P_e + (1 - P_e) \cdot P_{CE} + P_e \cdot P_{EC}))^m \quad (5)$$

where m is the size of block. Then, we express the total number of transmitted blocks K as:

$$K = \lceil n \cdot P'_B \rceil + n \quad (6)$$

If a SV-threshold is selected too aggressively (overly high), the priorities of "clean" blocks would be incorrectly reduced, i.e., P_{CE} would increase in Eq. (5). On the other hand, if a threshold is set to be too low, it would not be sufficiently powerful to detect blocks that are received in error, i.e., P_{EC} would increase in Eq. (5). Therefore, in order to maximize the throughput performance, we should choose an optimal *SV-threshold* that achieves minimum P'_B. We evaluate the impact of the selection of SV-threshold via simulations. Fig. 11 shows the performance of Drizzle with different SV-thresholds under different bit error rates of the wireless channel. We use 4 different SV-thresholds to check how SV-threshold selection affects the performance. As shown in the figure, choosing a threshold as 27.5% gives the best performance among the four choices we have simulated. When higher thresholds are used (such as 52.5%, or 77.5%), the throughput is reduced, which indicates that overly aggressive screening may incorrectly reduce the priorities of "clean" blocks. On the other hand, we observe that a threshold that is too low (such as 2.5%) also negatively affects throughput performance, as "dirty" blocks remain in the set used for decoding. This observation shows that we should carefully tune the SV-threshold. Heuristically, 22% is used in Drizzle based on a large number of simulations that we have performed.

The *second* function of soft decision values in Drizzle is to count the number of "dirty" blocks in the set used for decoding. When the decoding fails after each transmission, the receiver will count the number of "dirty" blocks in the decoding set, and ask the sender to transmit the same number

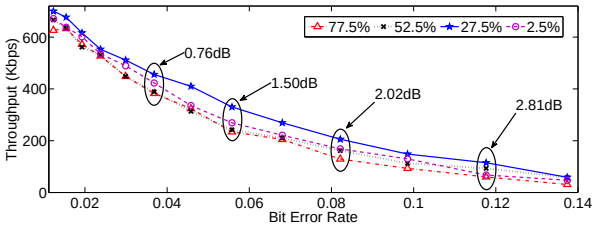


Fig. 11. The selection of SV-thresholds affects the performance of Drizzle. The performance of Drizzle under 4 different SV-thresholds, 77.5%, 52.5%, 27.5%, and 2.5%, is evaluated to show the importance of SV-threshold selection. Values in dB are the gains that the best case outperforms the worst case in the simulation. Simulations are performed with the settings provided in Sec. VII.

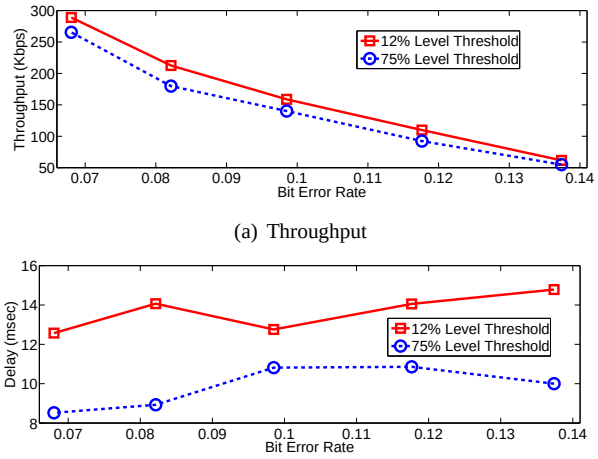
of additional coded blocks. This number will determine the number of blocks retransmitted, and will directly affect the performance of Drizzle. As such, a threshold must be set in Drizzle (referred to as level-threshold), so that blocks with confidence levels lower than this threshold will be counted as "dirty" ones. Let us denote the number of transmitted coded blocks after i^{th} transmission as d_i. Then, we have:

$$d_1 = n$$
$$d_i = d_{i-1} + \lceil d_{i-1} \cdot P_L \rceil \ , i = 1, \cdots, \overline{k}$$

where $\overline{k}$ is the maximum number of retransmission and P_L is the probability of counting a block as a "dirty" block. If a high level-threshold (large P_L) is selected, correctly received blocks could be counted as error blocks and extra retransmissions will be requested (large d_i), which will consume more bandwidth. Since more redundant retransmission blocks are transmitted, it is with a higher possibility to correctly recover the packet in the receiver at the next retransmission (a small number of retransmissions, i.e., small i). On the other hand, if a low level-threshold is selected (small P_L), error blocks could not be detected. It will also cause extra retransmission for error correction after the failure of network decoding (large number of retransmissions, i.e., large i). In this case, whereas barely required blocks are re-transmitted, more retransmission requests are required, which cause delays.

Drizzle is designed to be able to adjust the level-threshold depending on specific requirements of the applications. If the application is delay sensitive (such as voice), the level-threshold should be set to be high in order to conservatively request more coded blocks (larger P_L) at the following transmission. Otherwise, if the application requires a higher throughput, the level-threshold could be set to be lower (smaller P_L), to request a barely sufficient number of additional coded blocks, so that available bandwidth can be most efficiently used. Fig. 12 shows the delay and throughput tradeoff of two different level-thresholds, with the values of 12% and 75% are used in this simulation. A level-threshold of 12% shows 14% higher throughput on average, whereas a level-threshold of 75% is 39% better with respect to delays on average.

Theoretically, SV-threshold and level-threshold can be dynamically adjusted to adapt to the network environment, including channel quality, mobility, and the transmission mode. Another potential solution for threshold selection can be obtained by historical data learning. In our future work, we will further study the optimal thresholds for Drizzle using learning techniques.



(a) Throughput

(b) Delay

Fig. 12. The level-threshold affects the delay and throughput performance in Drizzle. A higher level-threshold is helpful to achieve higher throughput, but with a larger delay. On the contrary, a lower level-threshold leads to lower throughput, but with smaller delays. Simulations are performed with the settings provided in Sec. VII.

C. How Does SV Work in Cooperative Transmission?

By applying *normalized SV* and *adaptive threshold* described above, Drizzle is able to check the correctness of each coded block it collects, no matter where the block comes from and which modulation scheme is used on it. *Normalized SV* and *adaptive threshold* techniques in Drizzle are essentially a way to perform link adaptation by tightly integrating with the adaptive modulation scheme adopted in the physical layer of multi-channel wireless networks. They are especially helpful to achieve *cooperative transmission* in Drizzle as described in Sec. III-C. Although, in the multi-path transmissions, different senders may use different modulation schemes to transmit coded blocks to the same receiver, as channel conditions are different on each path (adaptive modulation is applied), the receiver could check the correctness of all the blocks effectively by applying *normalized SV* and *adaptive threshold*. Drizzle makes it possible for the receivers to correctly select clean blocks and decode the original packets successfully.

We perform simulations to examine the effectiveness of Drizzle in such a multi-path transmission scenario. In the simulation, two upstream nodes serve as senders and transmit data to the same receiver via two separate sub-channels, and QPSK and 16QAM are used on each path respectively. By applying adaptive modulation, the modulation schemes are determined to meet the target Bit Error Rate (BER) based on the estimated SNR. For example, if the target BER is 10^{-3}, 16QAM is used at a SNR of 8dB and 64QAM is used at a SNR of 12dB. Now, we set the modulation schemes as QPSK and 16QAM in the simulation. Then, we calculate the SNRs on each channel according to the target BERs (by adopting the solution in [29]). By varying the target BERs, we examined the downlink throughput at the receiver under different channel conditions (with different SNRs). Fig. 13 shows the simulation results, where a 3.24dB gain can be achieved on average by

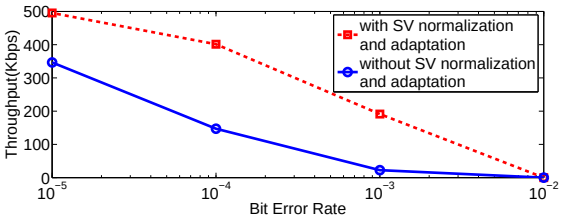


Fig. 13. A comparison of Drizzle's performance with and without SV normalization and adaptation in a cooperative transmission scenario. Two nodes are sending coded blocks to one receiver using different modulation schemes (QPSK and 16QAM are used on each sender respectively). The transmission is under different channel qualities (SNRs), which are generated by varying the target BERs in a certain range. Simulations are performed with the settings provided in Sec. VII.

applying *normalized SV* and *adaptive threshold*. This shows a significant benefit when link adaptation in Drizzle is applied.

V. Implementation Issues in Drizzle

As Drizzle uses network coding at its core, we are aware of a few implementation issues that, if not appropriately addressed, may affect the performance.

A. Choosing a Size for Coded Blocks

As we have shown, each packet is divided into a number of blocks, on which random network coding is performed. At a glance, it may appear that a smaller block is always preferable, as a smaller block lead to less overhead when retransmissions are made, and more accurate confidence levels as the normalized SVs of bits are averaged.

Unfortunately, a block that is too small will lead to an inherent problem that is hard to address. A block with m bits has to use at least GF(2^m) to perform random network coding, and a smaller number of bits in a block leads to a smaller size of the Galois Field with a smaller degree of freedom when coefficient vectors are randomly chosen. This leads to a higher probability of producing linearly dependent blocks with random network coding.

It is therefore important to choose an appropriate size for coded blocks, so that a block is sufficiently small, but supports a sufficient degree of freedom to generate randomized coefficient vectors that are linearly independent of one another. We have studied how the selection of block sizes affects the performance of Drizzle through both analysis and simulations as follows.

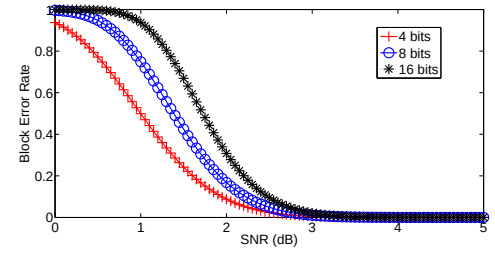
The block error rate (P_B) can be stated as:

$$P_B = 1 - (1 - P_e)^m \tag{7}$$

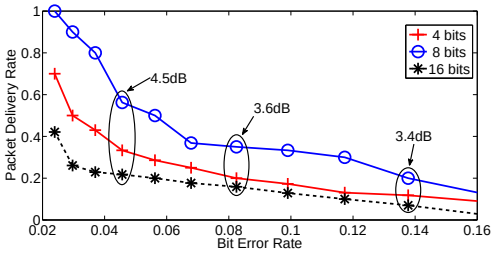
where P_e is the bit error rate, and m is the block size in bits. Then, the packet delivery rate (P_S) considering decoding errors can be expressed as the following:

$$P_S = \sum_{j=n}^{K} \binom{K}{j} (1 - P_B)^j P_B^{K-j} (1 - P_D) \tag{8}$$

where j is the received number of blocks without error, K is the total number of transmitted blocks, n is the batch size and P_D is the decoding error probability due to linearly dependent random coefficients. P_D decreases as the block size increases,



(a) Block Error Rate



(b) Packet Delivery Rate

Fig. 14. The selection of block sizes impacts the performance of Drizzle. (a) The performance of block error rates under 3 different block sizes: 4 bits, 8 bits and 16 bits. (b) The performance of packet delivery rates ($K = 2n$) under a Rayleigh fading channel. Simulations are performed with the settings provided in Sec. VII.

because a larger field size provides a larger degree of freedom in randomly chosen coefficient vectors.

In order to show the effect of different block sizes to the packet delivery rate, we consider a packet with a size of 512 bits which is divided into 128, 64, 32 blocks for $m = 4, 8, 16$, respectively. Block error rates and packet delivery rates are shown in Fig. 14. We can clearly see that as the block size increases, the block error rate also increases. However, due to the high decoding error probability with small block sizes, its packet delivery rate, which is a ratio of the number of error-free packets over the total transmitted packets, suffers from poor performance. Considering the tradeoff between the block error rate and the decoding error probability generated by blocks with different sizes, we select 8 bits as the best tradeoff.

In our simulation concerning the packet delivery rate, an 8-bit block size shows the best tradeoff with $K = 2n$, $K = 3n$, and $K = 4n$. Simulation results of packet delivery rates with $K = 2n$ are shown in the figure. Thus, we adopt GF(2^8) to perform random network coding. In this case, a block may contain multiple symbols when a symbol is smaller than 8 bits. For example, a block contains 4 symbols with QPSK modulation, where a symbol has 2 bits. In 16QAM modulation, a block includes 2 symbols. Our simulation results shown in Sec. VII have further verified the effectiveness of our choice of the block size.

B. Reducing the Overhead of Carrying Coefficients

In Drizzle, it is important to reduce the overhead of communicating random coefficients from the sender to the receiver for each coded block. Since the size of the block is small, the number of blocks in a packet will be large (64 blocks in a 512-bit packet, for example) with a large number of corresponding coefficients. Regardless of how we carry these coefficients, the overhead over wireless channels will be prohibitive.

Our solution is to avoid the communication of coefficients between the receivers and senders. In Drizzle, the random coefficient matrix is pre-generated and kept at both the senders and the receivers. In WiMAX, each sender-receiver pair needs to negotiate parameters such as modulation, coding, and transmission power, before the actual data transmission. In Drizzle, the sender transmits the index of the pre-generated random coefficients matrix that is used for encoding to the receiver, as a part of the session control information (in HARQ, the session number is also communicated as a part of the session control information). In order to reduce the overhead of storing different coefficient matrices for different batch sizes and different maximum number of retransmission blocks, only one coefficient matrix with a minimum sufficient size is stored and used for encoding and decoding. Let us denote the maximum batch size as N, the maximum number of retransmission blocks as D, and the maximum number of cooperating upstream nodes as C. Then the dimension of the stored matrix is $N \times M$, where $M = N + D \times C$. To guarantee successful decoding, any $N \times N$ sub-matrix is produced to be nonsingular.

How can the reliability of index negotiation be ensured? Wireless systems like WiMAX and 3GPP employ a reliable management/control message transmission mechanism. Session control information is conveyed using management/control messages, which are protected by the modulation and coding scheme (MCS) level, which is more robust than regular data burst transmission, or reliable error control schemes using HARQ or MAC level ARQ. The Multiple-input and multiple-output (MIMO) antenna scheme and low density parity check (LDPC) codes use the same concept to transmit the index for pre-coding matrix and matrix H, respectively, which are pre-generated and kept in the transmitters and receivers. Therefore, the coefficient index can be effectively protected and guaranteed to be successfully distributed to both receivers and senders.

Upon receiving such an index, the receiver has full knowledge of all coefficient vectors used in future coded blocks from the sender, by looking up the pre-generated matrix. It is also possible to use a seed of a pseudo-random number generator instead of the index to specify a future sequence of coefficient vectors to be used by the sender.

C. Computational Complexity and Protocol Overhead

As neither base stations nor relay stations have constraints with respect to the energy and computational power, we are only concerned with the computation overhead at mobile stations. Modern mobile devices, such as smartphones, have abundant memory and computational power. According to the results in [30], random network coding is almost "free" with modern mobile processors. The coding speed could reach 1248 Mbps for 16 blocks of 32 KB each and 348 Mbps for 64 blocks of 32 KB each. As our block size is as small as a few bits, encoding and decoding are even much faster. Although it indeed incurs additional computation to some extent, it keeps the overhead within practical limits.

Drizzle has a much smaller overhead compared with previous work, MIXIT [7], which also performs symbol-level network coding. In MIXIT, blocks are coded across packets and only on the correct symbols. Thus, the header has to include several runs of random coefficients, which generates a large amount of overhead. Assume that the packet size is 1500 bytes (a typical size in IEEE 802.11 networks) and the batch size is 32 (typical number), with 4 runs. For each coded block, a 8.5% overhead is incurred, which is rather substantial. Usually, with 400 symbols, there are dozens of runs at the least, regardless of dynamic programming schemes used in MIXIT. Assume that there are 20 runs, which leads to a completely unacceptable overhead of 42%. Moreover, if the header is not correctly received, the decoding can not be performed (with most of the packets discarded). In contrast, Drizzle adopts a totally different approach by using a predefined codebook, which is only transmitted to the receivers once through reliable channels. There is no header overhead when coded blocks are transmitted.

Another problem in MIXIT is that the feedback (ACK/NACK) has to be reported to the sender via multiple hops through a shared channel, which may generate large delays, especially when the number of hops is large and the batch size is small. In Drizzle, the feedback information are transmitted via separate channels (control channels). Thus, the feedback messages are transmitted in parallel with the data, which do not generate any delay at all.

VI. Numerical Analysis

Beyond intuitive justifications, we seek to offer an in-depth understanding of the performance advantage of Drizzle as compared to HARQ, by developing analytical models for both Drizzle and HARQ with respect to throughput. For the sake of fairness, the same Modulation and Coding Scheme (MCS) is used on both protocols. To facilitate our mathematical analysis, we first define the following notations.

η_i	the SNR of each transmission
η_0	the SNR of the initial transmission
η	the effective SNR
r	the channel rate achieved by a MCS
$P_e(r, \eta_i)$	the bit error rate under given r and η_i
α	the ratio of redundancy packet size over the original packet size (for HARQ)
$\overline{k}$	the maximum number of retransmissions
$p(\eta_i)$	the error probability of ACK/NACK under a given η_i
T_{delay}	average transmission time of an ACK/NACK packet
T_{out}	timeout period in the ARQ scheme
T_{cap}	transmission time of one packet with the channel capacity
B	the number of successfully decoded bits
τ	the time slots occupied for the transmission

The average throughput under a certain channel rate r and SNR η, denoted as $T(r, \eta)$, can be defined as the ratio of the expected number of successfully decoded bits $E(B|r, \eta)$ to the expected time slots $E(\tau|r, \eta)$ occupied for the transmission, as shown in the following:

$$T(r, \eta) = \frac{E(B|r, \eta)}{E(\tau|r, \eta)} \tag{9}$$

A. Drizzle

As Drizzle only transmits a barely sufficient number of symbol-level blocks for each transmission, the throughput of Drizzle mainly depends on the block error rate. To calculate

it, we set that the receiver can correctly receive n (n can be the batch number) linearly independent coded blocks after the sender transmits a total of K coded blocks. We can represent K as follows:

$$\min\{K : \sum_{i=1}^{K}(1 - P_B(r,\eta_i))(1 - P_D) \geq n\} \quad (10)$$

$P_B = 1 - (1 - P_e(r,\eta_i))^m$ is the block error rate in Drizzle, where m is the number of bits in one block. P_D is the decoding error rate due to the linear dependence of random coefficients. Therefore, the throughput of Drizzle can be calculated as:

$$T_{Drizzle}(r,\eta) = \frac{n}{K} \cdot r \quad (11)$$

B. HARQ

In [31], it has been indicated that a valid approach for modeling the soft combining in HARQ is to simply add the SNR value after each combining process. Based on that, a tractable model was proposed in [31], and extensive simulation results in [32] also support the model. Cho *et al.* [33] slightly modified the model and concluded that the ratio of the SNR increment per retransmission to the original SNR, *i.e.*, $\Delta\eta/\eta_0$, is proportional to α: $\Delta\eta = g\alpha\eta_0$, where the IR coding gain $g \geq 1$ is weakly dependent on the modulation and coding scheme [31], [34], [32]. We adopt this basic model.

We seek to provide a general formulation for the HARQ throughput, which could represent its performance no matter which coding schemes are used. First, the expected number of successful decoded bits per time slot is calculated by:

$$E(B|r,\eta) = r\big(1 - \prod_{j=0}^{\overline{k}} P_p(r,(1+jg\alpha)\eta_0)\big) \quad (12)$$

where P_p is the packet error rate. We assume one packet contains l blocks in Drizzle. Thus, we have,

$$P_p(r,\eta) = 1 - (1 - P_B(r,\eta))^l \quad (13)$$

We assume that the receiver is able to decode the packet correctly after the kth retransmission. The probability for that is:

$$P_t(k) = \big(1 - P_p(r,(1+kg\alpha)\eta_0)\big) \cdot \frac{\prod_{q=0}^{k} P_p(r,(1+qg\alpha)\eta_0)}{P_p(r,(1+kg\alpha)\eta_0)} \quad (14)$$

The number of time slots for the initial packet and redundancy retransmissions is given by:

$$T_{\text{DATA}} = 1 + k\alpha \quad (15)$$

Recall that the effects of ACK/NACK overhead need to be taken into consideration. The delays caused by such overhead include the ACK/NACK packet transmission time and the delay due to ARQ timeout (when ACK/NACK packets are lost and the sender has to wait for an ARQ timeout to retransmit). Considering the probabilities of both cases, the number of time slots of such ACK delay is given by:

$$T_{\text{ACK}} = k\Big\{p(\eta_0)\frac{T_{out}}{T_{cap}} + (1 - p(\eta_0))\frac{T_{delay}}{T_{cap}}\Big\} \quad (16)$$

We now consider the situation where the packet has been correctly decoded after the kth retransmission, but the ACK corresponding to this successful transmission is lost. Further delays may ensue because the sender will transmit additional redundancy packets under this situation. We can compute such delays by considering two cases: (1) the delay when the sender finally receives the ACK after the sth retransmission; and (2) the delay when the sender fails to receive any ACKs before the maximum number is reached. Considering the probabilities of both cases, the number of time slots is:

$$\begin{aligned} T_{\text{Timeout}} = \quad & \sum_{s=k}^{\overline{k}} \Big\{(s-k)(\alpha + \frac{T_{out}}{T_{cap}}) + \frac{T_{delay}}{T_{cap}} \\ & \times p(\eta_0)^{s-k}(1 - p(\eta_0))\Big\} \\ & + (\overline{k} - k + 1)(\alpha + \frac{T_{out}}{T_{cap}})p(\eta_0)^{\overline{k}-k+1} \end{aligned} \quad (17)$$

What if the packet could not be correctly decoded after the maximum number of retransmissions is reached? At first, the probability for this situation is:

$$P_{\max} = \prod_{j=0}^{\overline{k}} p_p(r,(1+jg\alpha)\eta_0) \quad (18)$$

The number of time slots for the data transmission in this case:

$$T_{\max} = 1 + \overline{k}\alpha \quad (19)$$

and the NACK feedback delay in this case can be computed as:

$$T_{\text{NACK}} = (\overline{k}+1)\Big\{p(\eta_0)\frac{T_{out}}{T_{cap}} + (1 - p(\eta_0))\frac{T_{delay}}{T_{cap}}\Big\} \quad (20)$$

The total expected number of time slots consumed now becomes:

$$\begin{aligned} E(\tau|r,\eta) = \quad & \sum_{k=0}^{\overline{k}} \Big(T_{\text{DATA}} + T_{\text{ACK}} + T_{\text{Timeout}}\Big)P_t(k) \\ & + (T_{\max} + T_{\text{NACK}})P_{\max} \end{aligned} \quad (21)$$

Finally, the throughput of HARQ can be derived by substituting Eq. (12) and Eq. (21) into Eq. (9). We note that the number of retransmissions should be kept within bounds in practical implementations of WiMAX, since unlimited retransmissions are not desirable with respect to delay. On the other hand, however, if the maximum number of retransmissions is too limited, packets may become lost, and such packet loss will affect *transmission continuity*, defined as the ratio of the number of dropped packets to the number of data packets that the sender transmits (excluding retransmissions). The probability of losing a packet in HARQ can also be easily derived:

$$P_{loss} = \prod_{j=0}^{\overline{k}} p_p(r,(1+jg\alpha)\eta_0) \quad (22)$$

From our analytical models, we can clearly see that HARQ would generate a substantial amount of overhead, while Drizzle efficiently utilizes the bandwidth to transmit a sufficient number of bits to decode the original packets.

We further show our numerical results to evaluate the performance of Drizzle and HARQ. In the evaluation, we apply conventional turbo codes, which have been employed in WiMAX. The bit and packet error rates in additive white Gaussian noise (AWGN) model are obtained through extensive simulations based on the technical specification document

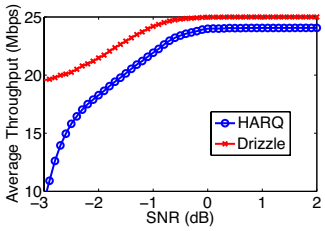


Fig. 15. The performance of Drizzle and HARQ based on numerical evaluation.

[35]. With respect to parameter settings in our simulations, the ARQ retransmission timeout period is set to be 0.05 milliseconds. In HARQ IR, we set the maximum number of retransmissions to be 4, and the corresponding optimal size of redundancy packets based on the results in [33]. We set the packet size as 512 bits and the block size as 8 bits.

Fig. 15 shows the numerical throughput performance of Drizzle and HARQ in single-hop transmission. Clearly, Drizzle outperforms HARQ.

VII. Performance Evaluation

We are now ready to resort to extensive simulations to study Drizzle's performance. For this purpose, we take advantage of the latest communication toolbox in MATLAB for simulation implementation. MATLAB is efficient for evaluating the performance of physical-layer protocols, and it is well designed to simulate physical layer designs in multi-channel wireless networks with fading channel characteristics, modulation, and soft decision values. To be realistic, we evaluate Drizzle's performance in WiMAX networks, where the practical settings of a real-world WiMAX network configuration are adopted.

A. Simulation Settings

In our simulations, WiMAX networks are simulated according to typical parameters defined in the IEEE 802.16 standard [1] and WiMAX system evaluation methodology released by WiMAX forum [36]. The simulation parameter settings according to these two documents are listed in Table I. In particular, we have used mobility patterns that reflect realistic parameter settings in a practical wireless environment. To evaluate the performance, we compare Drizzle with HARQ (the predominant error control protocol in WiMAX and LTE), and SOFT from previous work [18] proposed in the setting of IEEE 802.11 networks. With respect to HARQ, we adopt the type-II HARQ which performs *Packet Soft Combining* in transmissions and employs *Viterbi Soft Decision Decoding* using soft decision values. In the multi-path transmission scenarios, maximal ratio combining is performed in HARQ. With respect to SOFT, we have simulated the protocol to the best of our knowledge according to all the available details presented in [18]. We focus on three typical communication scenarios of WiMAX: single-link transmissions, handovers and multi-hop transmissions on both uplink and downlink.

B. Drizzle in Single-link Transmissions

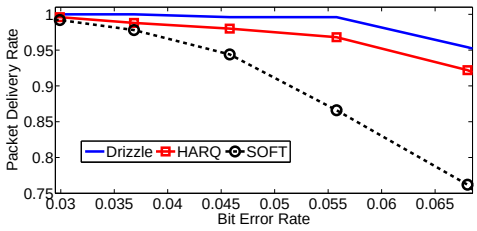
As a starting point, we first evaluate the performance of Drizzle in a basic, single-link transmission scenario. We perform the simulation that all three protocols are used to transfer a large file between a base station (BS) and a mobile station (MS) in the downlink. In this experiment, we are interested in two performance metrics: the packet delivery rate (calculated as the fraction of transmitted packets that are correctly delivered to the receiver), and the throughput. Fig. 16 shows a performance comparison among Drizzle, HARQ and SOFT under various BERs. The performance with respect to packet delivery rates is shown in Fig. 16(a), and Fig. 16(b) shows the corresponding throughput for all three protocols. From the results, we could easily observe that Drizzle's packet delivery rate (average: 0.99) is higher than both HARQ (average: 0.97) and SOFT (average: 0.91) by 1.89% and 9.14% respectively. The performance gain becomes more substantial when throughput is considered. Drizzle outperforms HARQ and SOFT by 33.6% and 55.8%, respectively. This is because Drizzle is designed to tightly integrate with the WiMAX physical layer for efficient bandwidth utilization. Due to the fact that Drizzle utilizes scarce bandwidth very efficiently by transmitting a barely sufficient number of symbols to recover the error packet, as discussed in Sec. III-B, a small performance gain in packet delivery rate can result in a large throughput performance gain. These improvements are supported by the efficient use of available wireless bandwidth, due to adaptive retransmissions in Drizzle.

Although the observed performance improvement is quite

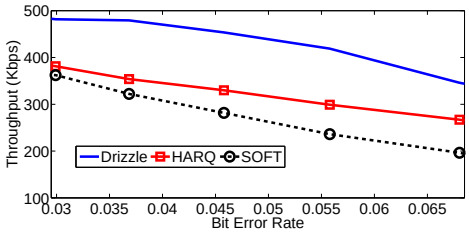
Channel Type	Rayleigh fading channel and AWGN
Path loss Model	COST-HATA-231[a]
Sampling time	0.1 second
Transmitter Power (Base Station)	25 dBm
Transmitter Power (Mobile Station)	20 dBm
Noise Power	-129.5 dBW
Packet Size	512 bits
Number of Blocks in a segment	64
Adaptive Modulation	used
OFDMA	used

[a]The extended HATA model to 2GHz by the European Cooperative for Scientific and Technical (COST) research.

TABLE I
SIMULATION PARAMETERS.



(a) Packet Delivery Rate



(b) Throughput

Fig. 16. Packet Delivery Rate and Throughput with a range of BERs.

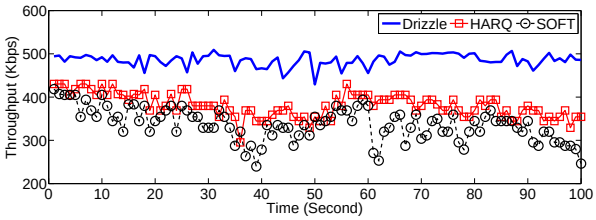


Fig. 17. Throughput in a single, time-varying wireless link with mobility.

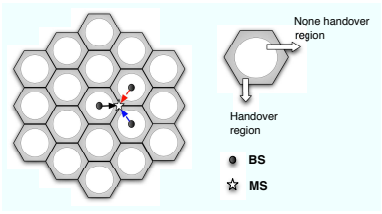


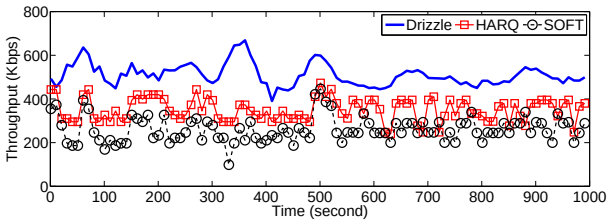
Fig. 18. Simulation setup in the WiMAX handover scenario.

encouraging under stable channel conditions, we focus more on the performance under realistic wireless environments with fluctuating channel conditions. To evaluate the performance of all three protocols, we run the simulation under the following scenarios. One MS moves around the service area of a cell randomly. Its initial speed (in km/h) and direction (in degrees) are generated with a uniform distribution of $U[10, 80]$ and $U[0, 360]$, respectively. The MS will change its speed and direction after a certain amount of time with an exponential distribution, with a mean value of 10 seconds. The new speed is uniformly generated with $U[10, 80]$ if the current speed is below 10 km/h; otherwise, it is obtained using $U[v - 10, v + 10]$, where v is the current speed. The new direction is obtained from a Gaussian distribution with the mean as the current direction, and a standard deviation of 40 degrees. The initial location of MS is randomly chosen in the service region. The design of this simulation scenario aims to provide realistic time-varying channel conditions. Moreover, we apply multi-path Rayleigh fading in the transmission, since the MS keeps on moving.

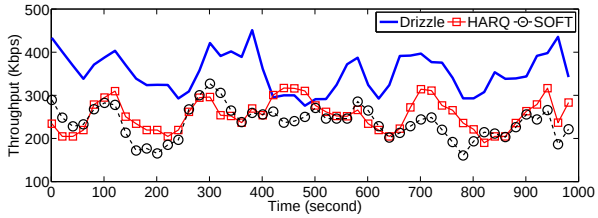
Fig. 17 shows the downlink throughput performance of all three protocols. We observe from the results that Drizzle's throughput (average: 485.95 Kbps) performs substantially better than both HARQ (average: 381.27 Kbps) and SOFT (average: 336.95 Kbps). This observation coincides with our intuition and is not a surprise: it shows Drizzle's ability to adaptively match its transmissions to the available bandwidth in time-varying channels, which helps in maintaining higher throughput.

C. Drizzle in the Handover Scenario

We next try to identify the potential performance gain offered by cooperative transmissions in Drizzle in the WiMAX handover scenario, as compared to HARQ and SOFT. Our evaluation is performed under the following scenarios. A total of 19 BSs are deployed in the service area. The cell sites are laid out as shown in Fig. 18, in which the MS is allowed to move around in the service area as the same fashion in the single-link case. At the handover region, the MS is able to enjoy the multi-path communication and perform *cooperative transmission*. Fig. 19 shows both uplink and



(a) Downlink throughput



(b) Uplink throughput

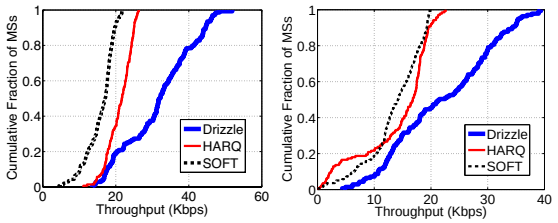
Fig. 19. Throughput comparison in the WiMAX handover scenario.

downlink throughput at the destination for all protocols from 1000-second simulations. The average throughput results are: 505 Kbps (downlink) and 352 Kbps (uplink) for Drizzle, 351 Kbps (downlink) and 303 Kbps (uplink) for HARQ, 259 Kbps (downlink) and 237 Kbps (uplink) for SOFT. In this scenario, the improvement with Drizzle reaches 44% and 95% over HARQ and SOFT, respectively, on downlink transmissions. At the same time, Drizzle outperforms HARQ and SOFT by 38% and 50% on the uplink. Such a throughput advantage should be considered substantial by any standard.

With the objective of becoming even more realistic, we seek to extend our performance evaluation to a large scale scenario. In the cellular system described previously, we set a large number of MSs active in the service region concurrently. The arrival process of new MS connections in each cell is assumed to be a Poisson process with a mean of 5 connections/cell/second. The MS active time duration is exponentially distributed with a mean of 100 seconds. Every active MS is moving around the service area using the same way as the previous simulation. We run the simulation for 1000 seconds, and the downlink throughput at the MSs is examined. From the results, there are a total of 95,010 MSs that have ever been active in the service area during the simulation time, with 460 MSs active simultaneously in each cell on average. Fig. 20 plots the CDF of the average throughput for both uplink and downlink transmissions, considering all active MSs in the simulation. Not surprisingly, Drizzle outperforms HARQ and SOFT by 50% and 100% respectively on the downlink with respect to the average throughput, due to its effective use of bandwidth and the advantages of random network coding in cooperative transmission. Further, Drizzle beats HARQ and SOFT by 56% and 62% respectively on uplink transmissions.

D. Drizzle in Multi-hop Transmissions

Finally, we illustrate the performance advantage of Drizzle, generated by both *adaptive retransmission* and *cooperative transmission*, in a WiMAX multi-hop transmission scenario. In order to extend the coverage area of a cell, relay stations (RSs) are placed within the border of the radio ranges of BSs.



(a) Downlink (b) Uplink

Fig. 20. Throughput performance in a large-scale handover scenario.

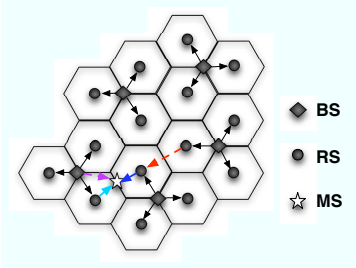
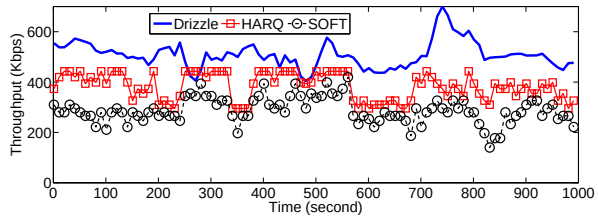


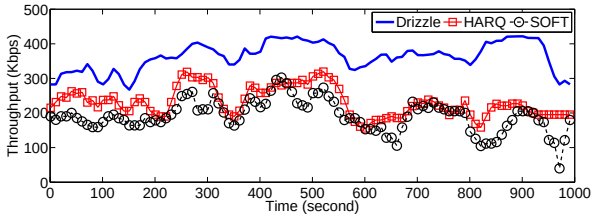
Fig. 21. Simulation setup in the WiMAX multi-hop scenario.

The simulation scenario is shown in Fig. 21, where a relatively large multi-hop network is considered. The MS in the simulation could communicate either directly with BS, or indirectly with BS through multiple hops connected by RS, as shown in Fig. 21 as an example. A similar evaluation is performed with the same setting as our simulation in the first handover case, where an MS randomly moves around and performs *adaptive retransmission* and *cooperative transmission* as long as such opportunities are explored. As shown in Fig. 22, we observe from the results that Drizzle obtains 32% and 77% average throughput improvement over HARQ and SOFT respectively on the downlink. The performance gains reach to 56% and 85% on the uplink. This demonstrates the ability of Drizzle to fully utilize available wireless spectrum in the multi-hop case.

Finally, we consider the case of a large-scale multi-hop network, with the same simulation setup as in the large-scale handover scenario. The maximum number of hops is limited to be 3. Fig. 23 presents the CDF of the throughput from 1000-second simulations. As expected, Drizzle outperforms HARQ and SOFT, and again by a substantial margin. In particular, Drizzle achieves a 80% higher throughput on average over

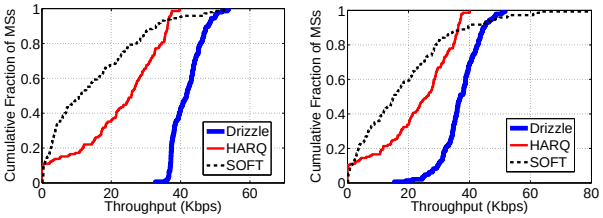


(a) Downlink throughput



(b) Uplink throughput

Fig. 22. Throughput in a realistic multi-hop case.



(a) Downlink (b) Uplink

Fig. 23. Throughput performance in a large-scale multi-hop scenario.

HARQ, as well as a 1.8x gain over SOFT in the downlink. Further, Drizzle performs better than HARQ and SOFT by 62% and 100%, respectively, in uplink transmissions. This confirms and highlights the benefits achieved by Drizzle in the multi-hop scenario, which is one of its design objectives.

VIII. Concluding Remarks

In this paper, we have explored the use of network coding in the physical layer of multi-channel wireless networks. Readers do not need to be reminded about the importance of studying multi-channel wireless networks using physical layer designs based on OFDMA: they represent the future generation of high-bandwidth wireless access technologies, as the momentum of both WiMAX and 3GPP LTE can demonstrate. Previous work in the literature has — almost without an exception — focused on IEEE 802.11 wireless networks with multiple hops, which uses a shared, single wireless channel, and a plain ARQ design in its MAC layer. Multi-channel wireless networks use Hybrid ARQ, which is able to retransmit additional redundancy to help successful packet decoding at the receiver. The highlight of this paper is our conclusion that, when network coding is used at the symbol level, Drizzle is able to outperform even HARQ, which is highly optimized in existing physical layer designs (such as WiMAX).

The intuition that Drizzle is able to outperform HARQ (not to mention existing work in 802.11-based networks) is quite simple to narrate: as its name implies, Drizzle allows the sender to retransmit a barely sufficient number of symbols that have not been successfully received at the receiver, and the receiver is able to hold the "bucket" until it is full of coded blocks, as if they are very fine "rain drops." Even better, the receiver can receive these blocks from more than one sender, with perfect collaboration across different senders, as multi-channel wireless networks create a large number of opportunities on multi-path transmissions. Since these "rain drops" are sufficiently small, there would be minimal waste of wireless bandwidth provided by the physical layer. As our extensive simulation results have shown, there is no surprise in our intuition: Drizzle is able to outperform both HARQ and related work in the literature by a substantial margin.

References

[1] "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems," *IEEE Std 802.16-2009 (Revision of IEEE Std 802.16-2004)*, pp. C1–2004, 29 2009.

[2] 3GPP TS36.300, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN): Overall Description."

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.
[4] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.
[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," in *Proc. of ACM SIGCOMM*, 2007.
[6] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in The Air: Practical Wireless Network Coding," in *Proc. of ACM SIGCOMM*, 2006.
[7] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-Level Network Coding for Wireless Mesh Networks," in *Proc. of ACM SIGCOMM*, 2008.
[8] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Application of Error-Control Coding," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 2531–2560, 1998.
[9] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding Over Routing in a Randomized Setting," in *Proc. of IEEE International Symposium on Information Theory, 2003.*, 2003.
[10] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "Soft Decision Metric Generation for QAM With Channel Estimation Error," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.
[11] D. Chase, "A Combined Coding and Modulation Approach for Communications Over Dispersive Channels," *IEEE Transaction on Communications*, vol. 21, pp. 159– 174, March 1973.
[12] E. Soljanin, N. Varnica, and P. Whiting, "Incremental Redundancy Hybrid ARQ with LDPC and Raptor Codes," *IEEE Transactions on Information Theory*, 2005.
[13] D.-H. Cho, J.-H. Song, M.-S. Kim, and K.-J. Han, "Performance Analysis of The IEEE 802.16 Wireless Metropolitan Area Network," in *Proc. of the 1st International Conference on Distributed Frameworks for Multimedia Applications (DFMA)*, 2005.
[14] F. Wang, A. Ghosh, R. Love, K. Stewart, R. Ratasuk, R. Bachu, Y. Sun, and Q. Zhao, "IEEE 802.16e System Performance: Analysis and Simulations," in *Proc. of the 16th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2005.
[15] J. Jin, B. Li, and T. Kong, "Is Random Network Coding Helpful in WiMAX?" in *Proc. of IEEE INFOCOM*, 2008.
[16] N. Mohamed, J. Al-Jaroodi, and H. Jiang, "Dependable user-level socket over dual networks," *J. Parallel Distrib. Comput.*, vol. 65, no. 10, pp. 1261–1270, 2005.
[17] K. Jamieson and H. Balakrishnan, "PPR: Partial Packet Recovery for Wireless Networks," in *Proc. of ACM SIGCOMM*, Kyoto, Japan, August 2007.
[18] G. R. Woo, P. Kheradpour, and D. Katabi, "Beyond The Bits: Cooperative Packet Recovery Using Physical Layer Information," in *Proc. of ACM MobiCom*, 2007.
[19] S. Katti, S. Gollakota, and D. Katabi, "Embracing Wireless Interference: Analog Network Coding," in *Proc. of ACM SIGCOMM*, 2007.
[20] S. Zhang, S. C. Liew, and P. P. Lam, "Physical-Layer Network Coding," in *Proc. of ACM MobiCom*, 2006.
[21] A. Yazdi, S. Sorour, S. Valaee, and R. Kim, "Reducing Symbol Loss Probability in the Downlink of an OFDMA Based Wireless Network," in *Proc. of IEEE ICC*, May 2008, pp. 3485–3489.
[22] R. Cogill, B. Shrader, and A. Ephremides, "Stability analysis of random linear coding across multicast sessions," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, July 2008, pp. 31–35.
[23] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding Over Routing in a Randomized Setting," in *Proc. of International Symposium on Information Theory (ISIT)*, 2003.
[24] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems on Information Transmission*, vol. 21, no. 1, pp. 1–12, Jan. 1985.
[25] M. M. Wang, W. Xiao, and T. Brown, "Soft Decision Metric Generation for QAM With Channel Estimation Error," *IEEE Transactions on Communications*, vol. 50, no. 7, pp. 1058–1061, July 2002.
[26] Z. Shen, J. Andrews, and B. Evans, "Optimal power allocation in multiuser OFDM systems," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, vol. 1, Dec. 2003, pp. 337–341 Vol.1.
[27] ——, "Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 6, pp. 2726–2737, Nov. 2005.
[28] I. Wong, Z. Shen, B. Evans, and J. Andrews, "A low complexity algorithm for proportional resource allocation in OFDMA systems," in *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, Oct. 2004, pp. 1–6.
[29] M. P. Fitz and J. P. Seymour, "On the bit error probability of QAM modulation ," in *International Journal of Wireless Information Networks*, 1994.
[30] H. Shojania and B. Li, "Parallelized Progressive Network Coding With Hardware Acceleration," in *Proc. of 5th IEEE International Workshop on Quality of Service (IWQoS)*, 2007.
[31] F. Frederiksen and T. E. Kolding, "Performance and Modeling of WCDMA/HSDPA Transmission/H-ARQ Schemes," in *Proc. of IEEE Vehicular Technology Conference (VTC)*, October 2002.
[32] P. Frenger, S. Parkvall, and E. Dahlman, "Performance Comparison of HARQ with Chase Combining and Incremental Redundancy for HSDPA," in *Proc. of IEEE VTC*, September 2001.
[33] J. W. Cho, Y. Chang, Y. Kim, and J. Chon, "Analytic Optimization of Hybrid ARQ Performance in Wireless Packet Data Systems," in *Proc. of IEEE GLOBECOM*, 2006.
[34] T. E. Kolding and F. Frederiksen, "Performance Aspects of WCDMA Systems with High Speed Downlink Packet Access (HSDPA)," in *Proc. of IEEE VTC*, September 2002.
[35] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE Standard 802.16: A Technical Overview of the WirelessMAN™ Air Interface for Broadband Wireless Access," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 98 – 107, 2002.
[36] WiMAX Forum, "WiMAX Forum™ WiMAX System Evaluation Methodology, Version 2.0," December 2007.

Ronny Yongho Kim received his B.S. degree in Department of Electronics Engineering at Inha University, Incheon, Korea, in 1998 and his M.S. degree from the Department of Electrical and Electronics Engineering at Yonsei University, Seoul, Korea, in 2003, where he is working toward Ph.D. degree. Since 1998, he has been with LG Electronics, where he is a senior research engineer working in the research and standardization of future wireless technologies, primarily within the area of MAC and network protocol design. He actively participated in IEEE 802.21 and made significant contributions during the development of IEEE 802.21-2008. He has been participating in IEEE 802.16 and making many contributions to the next generation of mobile WiMAX radio air interface. From 2005 to 2006, he served as a liaison of IEEE 802.16 and IEEE 802.21 between two working groups. His current research interests are mobility management, power efficient protocol, femtocell protocol, relay communication and network coding.

Jin Jin received his M.A.S. and B.Eng. both in Tsinghua University, Beijing, China. Since 2006, he is with the Computer Engineering Group in the Department of Electrical and Computer Engineering at the University of Toronto, where he is a Ph.D. candidate. His research interest mainly focuses on wireless networks, including WiMAX, WiFi, Cognitive Radio Networks. In particular, he applies a wide variety of theories, *e.g.*, network coding, stochastic processing, and optimization, in analyzing and designing algorithms for wireless networks.

Baochun Li received his B. Engr. degree in 1995 from Department of Computer Science and Technology, Tsinghua University, Beijing, China, and his M.S. and Ph.D. degrees in 1997 and 2000 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a Professor. He holds the Bell University Laboratories Endowed Chair in Computer Engineering since August 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. In 2009, he was the recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society. His research interests include large-scale multimedia systems, peer-to-peer networks, applications of network coding, and wireless networks. He is a senior member of IEEE, and a member of ACM.