Chorus: Collision Resolution for Efficient Wireless Broadcast

Xinyu Zhang and Kang G. Shin
Department of Electrical Engineering and Computer Science
The University of Michigan
Email: {xyzhang, kgshin}@eecs.umich.edu

***Abstract*— Traditional wireless broadcast protocols rely heavily on the 802.11-based CSMA/CA model, which avoids interference and collision by conservatively scheduling transmissions. While CSMA/CA is amenable to multiple concurrent unicasts, it tends to degrade broadcast performance, especially when there are a large number of nodes and links are lossy. In this paper, we propose a new, drastically different protocol called `Chorus` that improves the efficiency and scalability of broadcast service with a MAC layer that *allows* packet collisions. `Chorus` is built upon the observation that packets carrying the same data can be effectively detected and decoded, even when they overlap in time and have comparable signal strength. It performs collision resolution using symbol-level iterative decoding, and then combines the resolved symbols to reconstruct the packet. This *collision-tolerant* mechanism significantly improves the transmission diversity and spatial reuse in wireless broadcast, providing an asymptotic broadcast delay that is proportional to the network radius. This advantage is exploited further by `Chorus`'s MAC-layer cognitive sensing and scheduling scheme. We evaluate `Chorus` with symbol-level simulation, and validate its network-level performance via ns-2, in comparison with a typical CSMA/CA broadcast protocol.**

I. Introduction

Network-wide broadcasting is a fundamental communication primitive that serves as a building block for many other protocols in multi-hop wireless networks, such as route discovery and information dissemination. An efficient broadcast protocol needs to deliver a packet (or a continuous stream of packets) from the source node to all other nodes in the network, with high packet-delivery ratio (PDR) and low latency. To improve PDR when links are lossy, multiple relay nodes can forward and retransmit the packet, thereby creating retransmission diversity. To reduce latency and resource usage, however, the number of transmissions must be kept to minimum, since redundant retransmissions take up channel time, slowing down the packet's propagation to the edge of the network. Therefore, a delicate balance needs to be maintained between PDR and delay.

To date, efficient broadcast support, either theoretical analysis [1]–[3] or practical protocol design [4], has mostly focused on the CSMA/CA MAC-layer scheduling model. CSMA/CA has proven to be an effective distributed scheduling scheme, especially via the 802.11 family of MAC standards. The limitation of CSMA/CA, however, has not been examined carefully in broadcast protocols. While its fine-tuned sensing and scheduling scheme reduces collision, CSMA/CA inevitably loses transmission opportunities, lowering channel usage and spatial reuse. This problem is especially critical for network-wide broadcast with latency constraints.

Fig. 1(a) illustrates a typical scenario where CSMA/CA limits the broadcast efficiency. With CSMA/CA, at least three time slots are necessary to deliver one packet from source S to all other nodes. A and B cannot transmit concurrently, even if they have to forward the same packet. In a lossy network, suppose node D had already received the packet, while C and E await the retransmission from A and B, respectively. In an optimal scheduling protocol, A and B are allowed to transmit the packet concurrently, oblivious of the collision at D. However, this is not possible in CSMA/CA, as one of them will back off immediately upon sensing the other's activity.

The work described in this paper was supported in part by the NSF under Grant CNS-0721529.

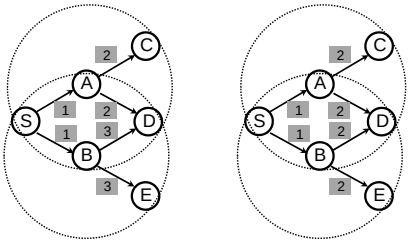

Fig. 1. Broadcast with traditional CSMA/CA in 802.11, in comparison with `Chorus`'s CSMA/CR (CSMA with collision resolution). The shaded tags denote the order of transmissions.

In this paper, we introduce a new broadcast protocol, called `Chorus`, based on a MAC layer that adopts CSMA with collision resolution (CSMA/CR). `Chorus` is built upon the key insight that *packets carrying the same data can be detected and decoded, even when they overlap at the receiver with comparable strength*. With `Chorus`, collision of the same packets from different relays can be effectively resolved. The advantage of such a collision-tolerant protocol is obvious, as shown in Fig. 1(b). With collision resolution, A and B can now transmit packets immediately and independently after receiving them from the source. Node D exploits `Chorus`'s collision resolution to decode the two collided packets from A and B. Therefore, only 2 time slots are required to deliver 1 packet over the entire network, due to the improved *spatial reuse*. Moreover, when links are unreliable, the two decoded packets from A and B create *transmit diversity* for the common receiver D, without consuming any additional channel time.

Both the spatial reuse and transmit diversity gain in `Chorus` are realized via its collision resolution scheme. Unlike traditional transmit diversity schemes such as beamforming [5], `Chorus` does not require symbol time synchronization nor instantaneous channel state information. In reality, it is infeasible to synchronize the independent transmitters A and B at symbol level [5]. `Chorus` exploits the asynchrony between them to identify collision-free symbols in the overlapping packets. It then initiates an iterative decoding process that subtracts clean and known symbols from collided ones, and obtains estimations of unknown symbols. The decoding succeeds as long as one packet has sufficient SNR, hence realizing the diversity offered by multiple transmitters.

At the MAC layer, `Chorus` adds a *cognitive sensing and scheduling* module to the 802.11 CSMA mechanism. Specifically, senders back off only when they sense a packet on the air that has a different identity from what they intend to transmit.

Such a cognitive MAC allows Chorus to fully exploit the advantage of collision resolution, while maintaining friendliness to background traffic. In addition, the collision-resolution capability enables anonymous broadcast at the network layer, without any topology or neighborhood information.

To quantify the effectiveness of Chorus, we establish an analytical framework for its achievable SNR and bit error rate (BER), which takes into account the error-propagation effects in iterative collision resolution. We further analyze its network-level performance in terms of latency and throughput. With a joint design of CSMA/CR and broadcast, Chorus achieves $\Theta(r)$ latency (r is the network radius), which is asymptotically lower than existing practical schemes.

To verify the feasibility of Chorus's collision resolution, we implement the iterative decoding and packet combination in a symbol-level simulator. To evaluate Chorus's performance in large networks, we feed the above fine-grained analytical and simulation results into the PHY layer of ns-2, implement the CSMA/CR MAC and broadcast protocol, and compare Chorus with a CSMA/CA based protocol. In a large set of randomly-chosen topologies, Chorus shows several-fold performance improvement in latency and PDR. The performance gain is relatively insensitive to network size, source rate and link quality, and is observed in both single- and multi-source broadcast scenarios. These properties are especially valuable for information dissemination in large-scale wireless networks, and signify the importance of exploiting PHY-layer signal processing to improve application performance.

The remainder of this paper is organized as follows. In Sec. II, we review existing work in contrast with Chorus. We introduce the collision resolution mechanism in Sec. III, and then the cognitive sensing, scheduling and network-layer broadcast scheme in Sec. IV. In Sec. V, we derive Chorus's achievable SNR and BER, and analyze its asymptotic broadcast performance. We evaluate Chorus's performance via simulation in Sec. VI, and conclude the paper in Sec. VII.

II. Related Work

Efficient broadcast in multihop wireless networks has been studied extensively, from both theoretical and practical perspectives. From the theoretical perspective, it is well-known that scheduling a minimum latency broadcast is NP-hard, either in a general undirected graph [3] or in a unit disk graph (UDG) [1]. Without the minimum latency constraint, analytical solutions demonstrated the feasibility of scheduling with time complexity $\Omega(r \log n)$ [6] in a distributed anonymous broadcast, and $r + O(\log r)$ [2] in centralized broadcast with known topology, where r and n denote the network radius and number of nodes. More recent work has improved the efficiency, and adopted more realistic models such as the interference graph [7].

Practical broadcast protocols have mostly adopted the 802.11 CSMA/CA and extended it to multi-hop networks. A main mechanism is to prune the topology, leaving only a backbone that covers the entire topology. The double-coverage broadcast [4], for example, reduces redundant transmissions by selecting nodes that cover more neighbors, while ensuring each node is covered at least twice, such that retransmission can be exploited to improve delivery ratio. The fundamental difference between Chorus and such existing protocols lies in its MAC layer scheduling protocol. With a joint design of CSMA/CR and network level broadcast, Chorus can achieve the $\Theta(r)$ latency bound, hence it has both theoretical and practical relevance.

The advent of high-performance software radios has been inspiring wireless protocols beyond the CSMA/CA paradigm. For instance, interference cancellation [8] can be used to resolve two collided packets with disparate strength. The main challenge in applying interference cancellation to multi-hop wireless networks is that the transmitters need delicate power control to ensure decodability. In Chorus, even two packets with similar strength can be effectively decoded, because each sees the other as a complement, rather than interferer. If the RSS of one packet is significantly lower than the other, such that it cannot be detected, then Chorus automatically resorts to the capture effect to decode the strong packet.

Chorus is partly inspired by the ZigZag protocol [9], which exploits the signal processing capability of software radios to solve the hidden terminal problem in WLANs. ZigZag extracts symbols from collided packets by identifying repeated collisions of two hidden terminals. It treats each collided packet as a sum over two packets. The two original packets are recovered from two known sums, similar to solving a linear system of equations. In the PHY layer, Chorus uses similar collision resolution mechanism as ZigZag, but it resolves multiple packets from a single collision, given that the packets are the same. In addition, Chorus aims at improving broadcast efficiency in wireless mesh networks, where it exploits transmit diversity and spatial reuse, using MAC layer cognitive sensing and broadcast scheduling.

The feasibility of allowing concurrent transmissions to create diversity has also been explored in communications. Concurrent cooperative communication [10], for example, allows co-located wireless nodes to transmit at the same time, thus forming a virtual antenna array that increases signal strength at the common receiver. Beamforming protocols [5] synchronize the transmitters, such that their signals can combine coherently at the receiver. These techniques require strict frequency, phase, and time synchronization at the symbol level, among distributed transmitters. Such fined-grained synchronization remains an open challenge [5], due to the limited time resolution at the wireless nodes, and the variation of the wireless channels.

III. Collision Resolution in Chorus

In this section, we introduce the physical-layer collision resolution in Chorus. For clarity, we start with a simple case of two-packet collision, focusing on how to detect, decode, and combine the collided packets to achieve the diversity gain. Then, we deal with the general case of resolving more than two packets' collision. Note that we have adopted a similar PHY layer in a separate paper [11] which presents a more comprehensive introduction to the implementation of collision resolution in software radios. Its objective is to realize non-orthogonal cooperative communications without tight synchronization among relays.

A. Detecting Collided Packets

In Chorus, a transmitter attaches a known random sequence to the beginning of each packet as a preamble. The receiver then uses a *matched filter* to detect the exact arrival time of this preamble. A matched filter is an optimal linear correlator that maximizes the SNR when correlating unknown signals with

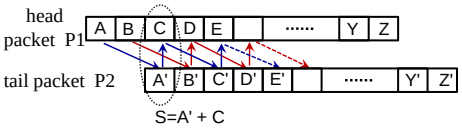

Fig. 2. Iteratively decoding two collided packets carrying the same data.

a known sequence [12]. It outputs a peak value whenever the packet preamble is detected, even if the preamble is hidden in a strong noise. It operates continuously, so that those preambles overlapping with other packets can still be identified. The number of preambles detected in a run indicates the number of overlapping packets at the receiver.

The peak output grows linearly with the number of bits in the preamble, and with the RSS of the packet [12]. Therefore, the detection threshold is also a linear function of these two factors [9]. In has been observed that using a 32-bit pseudo-random preamble, the collision detection probability is higher than 98% under practical wireless settings [9]. Hence, the preamble introduces negligible overhead to the packet.

B. Iterative Collision Resolution

Since a packet usually consists of thousands of symbols, the probability of two collided packets being aligned perfectly is close to zero. In practice, the higher-layer operations at transmitters introduce further randomness, resulting in asynchronous arrival time. We identify the natural offset between the two packets by detecting their preambles. Within the offset region, no collision occurs. We first decode the clean symbols therein, and then iteratively subtract such known symbols from the collided ones, thereby obtaining the desired symbol.

For instance, in Fig. 2, two packets (head packet P1 and tail packet P2) from different transmitters collide. We first decode the two clean symbols A and B in P1. Symbol C is corrupted as it collides with A' in P2, resulting in a combined symbol S. To recover C, note that symbols A' and A carry the same bit, but the analog forms are different because of channel distortion. Therefore, we need to reconstruct an image of A' by emulating the channel distortion over the corresponding bit that is already known via A. The channel distortion effects, including amplitude attenuation, phase shift, frequency offset, and timing offset, can be accurately estimated using standard communication techniques, as demonstrated in realistic experimental work [9].

After reconstruction, we subtract the emulated A' from S, obtaining a decision symbol for C. Then, the decision symbol is normalized using the channel estimation for P1, and a slicer decides if the bit in C is 0 or 1. For BPSK, the slicer outputs 0 if the normalized decision symbol has negative real part, and 1 otherwise. The decoded bit in C is then used to reconstruct C' and decode E. This process iterates until the end of the packet is reached. The iteration for other collided symbols proceeds similarly. The estimation, reconstruction and cancellation for higher-order modulation schemes, such as M-PSK (M=4, 8, 16, 64), can be realized in a similar way, except that the signal constellation is mapped to different complex numbers [9]. Also note that the above procedure has linear complexity with respect to packet length, which is similar to ZigZag [9] and interference cancellation [8].

Beside the iterative decoding in the forward direction, `Chorus` can also work backward, starting from the clean symbols in P2 (*i.e*, symbol Y' and Z'), until reaching its beginning, hence obtaining a different estimation of the packet. Chorus then performs the following packet combination to improve the decoding probability.

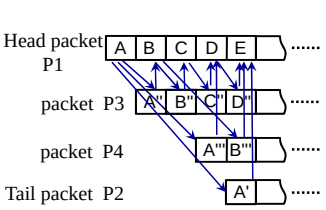

Fig. 3. Collision resolution: the multi-packet collision case.

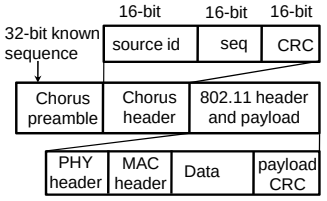

Fig. 4. The broadcast packet format in Chorus.

C. Using Packet Combination to Improve Diversity

Since P1 and P2 may have different strengths, their decoding confidence also differs. Decoding confidence is indicated by the magnitude of the decision symbol. The farther away it is from the decoding threshold (which is 0 in BPSK), the higher probability it can produce the correct bit, since this is equivalent to a higher SNR. Combining two decision symbols carrying the same bits (*e.g.*, A and A' in Fig. 2) can increase the decoding confidence. This is because the useful information is enhanced, while the noise within the two symbols is not combined coherently.

In Sec. V, we show that weighted summing over corresponding symbols can improve the decoding probability, when two versions of the same packets are received sequentially without collision. Such a weighted combination harvests full transmit diversity, *i.e.*, the SNR of the combined packet is the sum SNR of the two independently received packets.

For those iteratively decoded packets, we only use selective combination, *i.e.*, assigning weight 1 to the packet with the highest SNR, and 0 to all other packets. This is because a weighted combination over two iteratively decoded packets does not improve SNR. In fact, the iterative collision resolution in Chorus can cause error propagation, due to the correlation between consecutively decoded symbols. For example, in Fig. 2, if symbol A produces an erroneous bit, then the error propagates to A', which affects subsequent symbols such as C. Fortunately, such error propagation stops if the actual bits of A' and C are the same. In this case, after subtracting the error image of A', we obtain a strengthened symbol that indicates the correct bit of C. Error propagation also stops when symbol C has a much higher strength than A'. Based on these two intuitions, we bound Chorus' BER, proving that the probability of error propagation decays exponentially with the error length (Sec. V).

D. Multi-packet Collision Resolution

Since Chorus allows concurrent transmissions, multiple versions of a packet can collide, especially when the network has high density. The resolution of multi-packet collision is complicated by the fact that intermediate packets no longer have clean symbols at the beginning or end. Fig. 3 illustrates a typical scenario.

Denote the earliest and latest packets as *head packet* and *tail packet*, respectively. To decode the head packet, Chorus proceeds in a way similar to the two-packet case, except that it needs to subtract multiple reconstructed symbols, including the one from the tail and those from the intermediate packets.

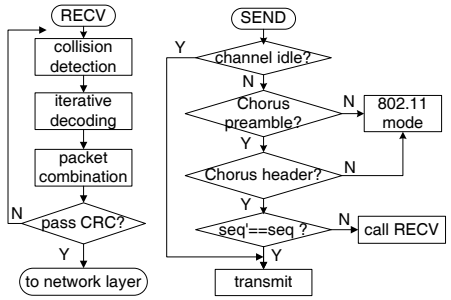

Fig. 5. The MAC layer control flow in Chorus. seq' denotes the sequence number of the packet on the air.

Similarly, another version can be obtained by decoding the tail packet, but in reverse order, starting from its end backward to the beginning. To obtain additional versions from intermediate packets, Chorus performs simple hard decoding. It tracks the packet symbol-by-symbol, treating all others as noise. Intuitively, the results have reasonable confidence only when this packet has much higher strength than others. The achievable decoding confidence will be rigorously characterized in Sec. V.

IV. Cognitive Sensing and Broadcast Schduling

Chorus's physical layer collision resolution must be integrated with the MAC layer, in order to reduce *unresolvable collisions* occurring when packets with different data collide. In addition, Chorus's network layer must ensure broadcast packets can reach the network edge. Next we detail both the MAC and network layer support for broadcast.

A. MAC Layer Cognitive Sensing and Scheduling

Chorus's MAC layer maintains the carrier sensing and backoff in the 802.11-based CSMA protocol, but adopts cognitive sensing that exploits the collision-resolution advantage, while avoiding unresolvable collisions. The principle of cognitive sensing is to decode the identity of the packet on the air, and accordingly, make the transmission decision. To this end, Chorus needs to add a new header field into the 802.11 packet.

1) Chorus packet format: Fig. 4 illustrates the broadcast packet format in Chorus. First, a known random sequence is attached to facilitate packet detection and offset identification (Sec. III-A). Second, a *Chorus header* field is added, which informs the receiver of the packet's identity, including the broadcast source's ID and the packet's sequence number. A 16-bit CRC (Cyclic Redundancy Check) [12] is included in this header. In case of CRC failure, this packet is discarded as it conveys wrong identity information.

When the headers of two packets collide, Chorus proceeds with the iterative decoding, assuming they have the same identity. After the decoding, it performs CRC over the header of each packet to ensure they are identical. If not, a decoding failure occurs, and both packets will be discarded. A decoding failure also happens when the CRC over the payload fails.

2) Scheduling of Sensing and Transmissions: With the collision-resolution capability, each transmitter calls a SEND procedure to perform cognitive sensing, as shown in Fig. 5 Transmitters make scheduling decision following three rules:

R1. Forward a packet immediately if the channel is idle.

R2. If the channel is busy, and the packet in the air is exactly one of the packets in the transmit queue, then start transmitting the pending packet.

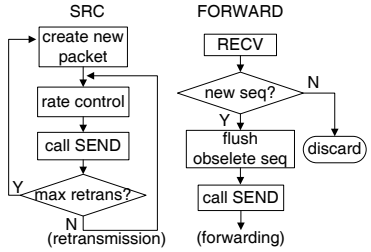

Fig. 6. Control flow for scheduling network-wide broadcast.

R3. If the channel is busy, but a preamble cannot be detected, or the header field of the packet on the air cannot be decoded, or a different packet is on the air, then start the backoff procedure according to the 802.11.

R1 is typical of all CSMA protocols. R2 is unique to the CSMA/CR-based scheme in Chorus. It enforces the principle behind Chorus, *i.e.*, overlapping packets carrying the same data may not cause collisions. Instead, by collision resolution, these packets offer transmit diversity to the receiver. Therefore, a sender node, such as node B in Fig. 1, can transmit its pending packet if it has the same identify as the one on the air (*e.g.*, the one that A is transmitting). In contrast, CSMA/CA transmitters stall and back off whenever the channel is busy.

R3 ensures friendliness to alien traffic, and is relevant for multi-source broadcast and co-existence with CSMA/CA based unicast traffic. To prevent unresolvable collisions between different packets, Chorus starts the normal 802.11 backoff if it senses that the channel is occupied by such alien traffic. To reduce interference to co-existing traffic, it also backoffs conservatively if the identity of the packet on the air cannot be decoded.

The advantages of cognitive sensing and scheduling come at the expense of additional overhead. In 802.11b, the sensing time slot is 20 μs, equivalent to the channel time of 20 bits in the broadcast mode. In contrast, Chorus needs to sense over the entire preamble and the header (80 bits in total, as indicated in Fig. 4). However, this overhead is negligible compared to the packet length. We will formalize the cost of the header overhead using both asymptotic analysis (Sec. V) and simulation experiments (Sec. VI).

B. Scheduling Network-wide Broadcast

Broadcast in Chorus is anonymous and decentralized. The source and relays do not need any topology information or neighbor identity. Following the SRC procedure in Fig. 6, the source node composes a Chorus packet, and transmits it like a normal 802.11 broadcast packet. *Any* neighbor who overhears this packet will provide best-effort service by forwarding it *once*, following the FORWARD procedure. Receivers with overlapped packets perform collision resolution before continuing with the packet relaying. After each successful reception, a receiver flushes those pending packets with obsolete seq, in order to prevent unresolvable collisions between packets with different sequence numbers. Intuitively, multiple versions of a packet proceed in parallel like a wavefront, which stops at the network edge. In case of continuous broadcast, the source node can control its rate to prevent congestion, and perform retransmission to improve PDR. These further optimizations are left to the application and will not be used in our evaluation.

When multiple broadcast sessions are running concurrently,

their packets are identified through the source-id field in the header part. Each relay maintains a transmit queue storing the packets to be forwarded. When the channel is idle, it directly transmits the head-of-line packet. Otherwise, it follows the MAC layer cognitive scheduling protocol, which maximizes the spatial reuse opportunity by scheduling the same packets, while avoiding collision with other broadcast sessions. Note that the co-existence with unicast traffic is a special case of multi-source broadcast. In effect, the latter case requires more conservative scheduling because of more severe interference, and therefore it will be used as a benchmark for validating Chorus's friendliness to alien traffic.

V. Performance Analysis

In this section, we first characterize the performance of collision resolution and packet combination in Chorus. We then analyze its asymptotic delay and throughput performance, in comparison with the traditional CSMA/CA schemes. The analytical results serve as guidelines for selecting the design parameters, such as packet-combining weights and maximum source rate.

Unless noted otherwise, we use the following set of notations: L for the packet length, F the offset between two collided packets, D the data rate, W the signal bandwidth, N the noise power, and δ^2 the noise variance. Multiple collided packets are indexed according to their arrival time, and γ_i denotes the SNR of packet i. We maintain consistent settings to the 802.11b broadcast mode. Specifically, all links adopt the 1Mbps basic access mode using BPSK [13] (assuming $D = 1$Mbps, $W = 1$MHz). No MAC-layer retransmission, ACK, RTS/CTS or other control packets are involved.

A. Achievable SNR and BER

We begin with an elementary scenario where two versions of a packet (denoted as P1 and P2) from different transmitters collide. This scenario is analogous to the two-user uplink channel in information theory [14], which adopts interference cancellation as the optimal decoder. However, Chorus's application scenario is unique in that P1 and P2 carry the same data. Ideally, they should complement, or at least do not interfere with each other. This intuition is formalized in the following set of theorems.

Theorem 1. *Without packet combination, the achievable SNR of* Chorus*'s collision resolution in the two-packet collision case is* $\Lambda = \max\{\frac{P_1}{N}, \frac{P_2}{N}\}$. *When decoding* m *overlapped packets, the achievable SNR of* Chorus*'s collision resolution is* $\Lambda = \max\{\frac{P_1}{N}, \frac{P_i}{\sum_{j\neq i} P_j+N}, \frac{P_m}{N}\}$, $i \in \{2, \ldots, m-1\}$.

Proof: The proof follows from Chorus's iterative decoding. We represent symbols in the complex form. Suppose at time t, symbol $\tilde{s}_1(t) = a_1 e^{j\theta_1} x_1(t)$ in P1 collides with $\tilde{s}_2(t) = a_2 e^{j\theta_2} x_2(t)$ in P2. Let v denote the receiver noise, then the received symbol $\tilde{s}(t) = \tilde{s}_1(t) + \tilde{s}_2(t) + v$. If we decode P1 first (forward-direction decoding), then $x_2(t) = x_1(t - F)$. In addition, the channel amplitude a_2 and phase θ_2 can be estimated via correlation, which can achieve high accuracy and introduces negligible noise [9]. Therefore, we can obtain a decision symbol for $x_1(t)$ as: $\tilde{s}(t) - \tilde{s}_2(t) = a_1 e^{j\theta_1} x_1(t) + v$. The resulting SNR level is: $\frac{|a_1 e^{j\theta_1}|^2}{2\delta^2} = \frac{P_1}{N}$, which equals the SNR when $s_1(t)$ is decoded independently.

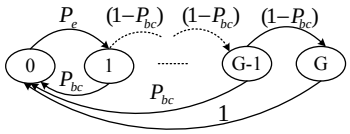

Fig. 7. The error propagation process as a Markov chain.

Similarly, if the clean symbols in P2 are decoded first (backward-direction decoding), then we can obtain $\frac{P_2}{N}$. Taking the maximum of these two yields $\Lambda = \max\{\frac{P_1}{N}, \frac{P_2}{N}\}$.

When m packets collide, the head and tail packets have clean symbols, and the achievable SNRs are $\frac{P_1}{N}$ and $\frac{P_m}{N}$, respectively, following a similar line of reasoning as above. Since Chorus performs hard decoding over intermediate packets, the achievable SNR for an intermediate packet is the same as treating other packets as noise, *i.e.*, $\frac{P_i}{\sum_{j\neq i} P_j+N}, \forall i \in \{2, \ldots, m-1\}$. The result follows directly after taking the maximum SNR of all packets. □

The above SNR bounds can be transformed to the BER bound that is directly related to the decoding performance [12]: BER $= Q(\sqrt{2\Lambda WD^{-1}}) = Q(\sqrt{2\Lambda})$. where the *Q-function* $Q(y) = \frac{1}{\sqrt{2\pi}} \int_y^\infty e^{-\frac{x^2}{2}} \mathrm{d}x$. $Q(y) \to 0$ exponentially when $y < 1$ and $y \to -\infty$, which also holds for $y > 1$ and $y \to \infty$. This implies that BER decreases exponentially with the achievable SNR.

B. Effects of Error Propagation

The above SNR and BER bounds are simplified in that they ignore the error propagation along sequentially-decoded symbols. Fortunately, the following analysis verifies that the error propagation has negligible effect in common cases.

We set up a Markov chain model that relates error propagation to the SNR of each packet, and the offset between collided packets. Again, we start with the two-packet collision scenario in Fig. 2 and analyze the iterative decoding of the head packet P1. As shown in Fig. 7, we define *states* according to the error propagation length, *i.e.*, the number of consecutive errors in a run. The state transition can be classified into two cases: (i) the probability that an independent decoding error occurs (transition from state 0 to state 1), which equals the BER of clean symbols in P1 (denoted as P_e), and (ii) the probability P_{bc} that error propagation stops, *i.e.*, the next bit is correct even when the current bit is erroneous. The probability of continuing error propagation is $1 - P_{bc}$. The maximum error propagation length starting from a clean symbol is $G = \lfloor \frac{L}{F} \rfloor$, since the distance between any two consecutively-decoded symbols equals F.

Obviously, this Markov chain is aperiodic and irreducible, and thus, the steady state distribution exists. Let π_i be the steady-state probability of state i, then we have the following balance equations:

$$\begin{cases} \pi_1 = \pi_0 \cdot P_e \\ \pi_i = \pi_{i-1} \cdot (1 - P_{bc}), i = 2, 3, \cdots, G. \\ \sum_{i=0}^{G} \pi_i = 1. \end{cases}$$

Solving for the steady state, we have:

$$\pi_0 = \left(1 + P_e \cdot (1 - (1 - P_{bc})^G) P_{bc}^{-1}\right)^{-1} \quad (1)$$

$$\pi_i = \pi_0 \cdot P_e \cdot (1 - P_{bc})^{i-1}, i = 1, 2, \cdots, G \quad (2)$$

We proceed to derive the probability P_{bc} that error propagation stops. BPSK symbols can be represented as real values

subject to channel attenuation, since decoding only depends on the in-phase part of the received symbol. Back to the example in Fig. 2, suppose symbol C carries bit "0" (mapped to -1 in BPSK), and the channel attenuation over C is X_a, then symbol C is represented as $-X_c$. Suppose symbol A' carries bit "1" (mapped to 1 in BPSK) with channel attenuation $X_{a'}$, then the collided symbol $S = -X_c + X_{a'} + v$, where v is the additive white Gaussion noise. In this case, `Chorus` should subtract $X_{a'}$ from S. However, if the estimation of symbol A is incorrect, it will propagate to C via A'. Specifically, `Chorus` erroneously subtracts $-X_{a'}$, resulting in a decision value $Y_c = -X_c + 2X_{a'} + v$. Similarly, when A' carries bit "0" but `Chorus` estimates it as "1" via A, the resulting decision value is $Y_c' = -X_c - 2X_{a'} + v$. A symmetric argument applies to the case when symbol C carries bit "1". Therefore, the probability that collision resolution outputs a correct bit is:

$$\begin{aligned} P_{bc} &= 0.5P\{Y_c' < 0\} + 0.5P\{Y_c < 0\} \\ &= 0.5P\{w < 2X_{a'} + X_c\} + 0.5P\{w < X_c - 2X_{a'}\} \quad (3) \end{aligned}$$

The first term in Eq. (3) can be bounded as:

$$\begin{aligned} &P\{w < 2X_{a'} + X_c\} = 1 - P\{w \geq 2X_{a'} + X_c\} \\ &\geq 1 - \delta^2(2X_{a'} + X_c)^{-2} \text{ (\textit{Chebyshev Inequality})} \\ &= 1 - (2\sqrt{2\gamma_2} + \sqrt{2\gamma_1})^{-2}. \end{aligned}$$

Both γ_1 and γ_2 are in normal scale, corresponding to practical log scale values ranging from 6dB and above [9]. Therefore, in the above equation, it is reasonable to assume $\gamma_1 \gg 1, \gamma_2 \gg 1$. Consequently, $P\{w < 2X_{a'} + X_c\} \approx 1$.

For the second term in Eq. (3), a closed-form estimation can be obtained:

$$\begin{aligned} \Gamma = P\{w < X_c - 2X_{a'}\} &= 1 - \frac{1}{\delta\sqrt{2\pi}} \int_{X_c - 2X_{a'}}^{\infty} e^{-\frac{u^2}{2\delta^2}} \mathrm{d}u \\ &= 1 - \frac{1}{\sqrt{2\pi}} \int_{\sqrt{2\gamma_1} - 2\sqrt{2\gamma_2}}^{\infty} e^{-\frac{z^2}{2}} \mathrm{d}z \ (note: z = \frac{u}{\delta}) \\ &= 1 - Q(\sqrt{2\gamma_1} - 2\sqrt{2\gamma_2}) \end{aligned}$$

In practice, since the two packets are from two different transmitters, the difference between γ_1 and γ_2 is larger than 1, even in dB scale. Given the exponential decaying of the $Q(\cdot)$ function (Sec.V-A), a practical estimation is $\Gamma \approx 1$ if $\gamma_1 \gg \gamma_2$ and $\Gamma \approx 0$ if $\gamma_1 \ll \gamma_2$.

Combining the analysis of the two terms in Eq. (3), we have $0.5 \leq P_{bc} \leq 1$, and P_{bc} transits fast from 0.5 to 1 when $\gamma_1 \ll \gamma_2$. This trend is also illustrated in Fig. 8.

Back to Eq. (1), we have $\pi_0 \leq (1 + P_e)^{-1} \approx 1 - P_e$. π_0 approximates this upper-bound as $G \to 1$, *i.e.*, the offset between the two packets approaches the packet size. Furthermore, in the common case $G > 1$, we have:

$$\pi_0 \geq (1 + P_e P_{bc}^{-1})^{-1} \geq (1 + 2P_e)^{-1} > 1 - 2P_e \quad (4)$$

Therefore, the bit error probability P_e' in iterative decoding is bounded as:

$$P_e \leq P_e' = 1 - \pi_0 < 2P_e \quad (5)$$

In practice, P_e is typically below 10^{-6}; the packet length is around 1KB. Hence P_e' has similar effect on packet error rate (PER) as P_e, even when it approaches the upper bound. This means *the effects of error propagation on PER is negligible*, which will be further verified in our bit-level simulation.

Combining the bounds for P_{bc} and P_e with Eq. (2), we conclude that *while resolving a given collision, the error propagation probability decays exponentially with the error length* (also shown in Fig. 9). This is consistent with the empirical observation in [9]. The above reasoning can be straightforwardly extended to multi-packet collision resolution, where the probability that error stops propagating is also close to or larger than 0.5, because previous erroneous bit may strengthen the current bit with probability 0.5.

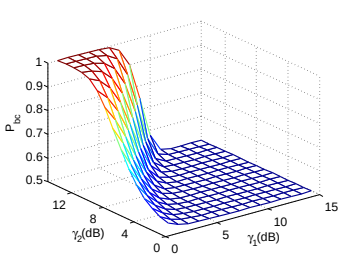

Fig. 8. Head packet's P_{bc}: the probability that error stops propagating to the next bit.

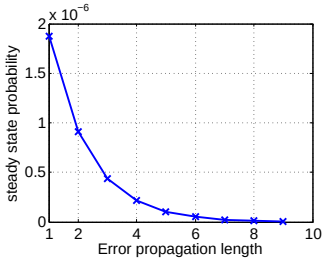

Fig. 9. Steady state distribution of error length. $\gamma_1 = 10, \gamma_2 = 7$. $F = \frac{L}{64}$. Error length 0 is not shown.

C. Optimal Packet Combination Weight

When two or more versions of the same packet are received sequentially without any collision, no error propagation occurs. Intuitively, this happens when a small packet size is used. Error propagation is also negligible when the two packets have a large offset F close to packet length. In such cases, we can harvest the transmit diversity via weighted combination of the symbols in the received versions. The optimal weight is derived as follows.

Theorem 2. *Without error propagation, the optimal combination weight of packet i is γ_i. The resulting SNR equals $\sum_{i=1}^{m} \gamma_i$.*

The proof is similar to the maximum ratio combining in multi-user communications [14], and is omitted due to space constraint. It should be noted that Theorem 2 does not hold when combining two or more iteratively-decoded packets with a small offset F, where error propagation occurs. In a high SNR region, the error propagation effect dominates the bit errors caused by noise, so the performance of the weighted combination can be worse than *selective combination*, *i.e.*, assigning weight 1 to the packet with the highest SNR, and 0 to all other packets. This intuition will be further justified via our simulation experiments.

D. Asymptotic Delay and Throughput

We now analyze `Chorus`'s network-level performance, including latency and throughput. To be consistent with existing asymptotic analysis [1], [2], [6], we assume perfect reception within the transmission range if no collision occurs. The network radius is r, *i.e.*, it spans r hops from the source to the receiver farthest away. Let h denote the size of `Chorus` preamble plus `Chorus` header, then we have the following asymptotic performance bound regarding broadcast latency and throughput.

Theorem 3. *The worst-case latency and throughput of* `Chorus` *is* $\frac{r(L+h)}{D}$ *and* $\frac{LD}{3(L+h)}$, respectively.

Proof: The network can be divided into r rings centered around the source node. A trivial lower-bound on the latency is $r\frac{L}{D}$, *i.e.*, all nodes within the same ring transmit concurrently after the previous ring, and the packet is repeated exactly r times. However, this is only achievable when the cognitive sensing

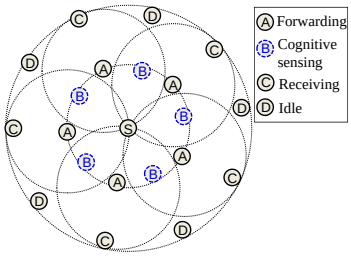

Fig. 10. The worst-case latency scenario in Chorus broadcast.

function is disabled. The worst-case scenario happens when cognitive sensing induces the longest delay between adjacent rings, as shown in Fig. 10. Specifically, at most a half of the nodes within each ring is transmitting while others within the same ring are transmitting. This induces latency equal to the duration of the Chorus preamble and header, which equals $\frac{h}{D}$. In addition, the latency can be repeated at most r times over the network, resulting in the worst-case latency $r\frac{L+h}{D}$.

In continuous broadcast, packets of different sequences must not collide as the collision cannot be resolved. To prevent such collisions, nodes within two hops cannot send different packets concurrently. Therefore, a new packet can be sent from the source only after the previous packets have propagated at least three hops away, which takes time $3\frac{L+h}{D}$. As a result, the amount of data transmitted within a unit time is: $\frac{L}{3\frac{L+h}{D}}$, which is equivalent to the broadcast throughput of Chorus. □

From Theorem 3, we see that the asymptotic latency of Chorus satisfies $\frac{rL}{D} \leq \Theta(r) \leq \frac{r(L+h)}{D}$. Under a unit disk graph model, Chorus's latency can be close to the trivial lower bound $\frac{rL}{D}$, since $h \ll L$. This is in sharp contrast with the $\Omega(r \log n)$ latency for anonymous broadcast using CSMA/CA [6].

Theorem 3 also reveals that the maximum supportable source rate (or maximum throughput) of Chorus is insensitive to the network size. As a worst-case bound, it can be used to control the source rate in continuous broadcast, in order to prevent the collision between consecutive packets and avoid congestion.

VI. Experimental Evaluation

We quantitatively evaluate the performance of Chorus in two steps. First, we use symbol-level simulation to verify the effectiveness of its collision-resolution scheme. Then, we introduce the implementation of Chorus based on the 802.11b module in ns-2, and evaluate its broadcast performance in large-scale networks. The simulation experiments further justify our previous analysis.

A. Collision-Resolution Performance

We implement a symbol-level simulator in Matlab. The symbols are represented as complex numbers, whose magnitude depends on the packet's SNR. We assume the receiver noise profile is AWGN, which is a typical approximation to the noise profile after receiver filtering and frequency compensation [9]. Given two or more collided packets, the simulator resolves the collision using Chorus's iterative decoding algorithm. The simulated receiver adopts a simple zero-forcing slicer which outputs a "0" bit if the decision symbol's real part is negative, and "1" otherwise. The signal bandwidth is set to 1MHz and data rate 1Mbps. The noise power density is 10^{-11} W/Hz. We vary the received signal power to simulate the SNR range

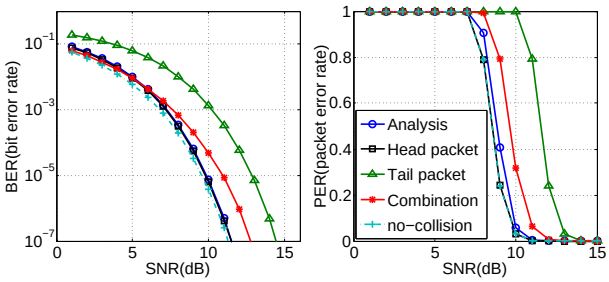

Fig. 11. Symbol-level simulation of iterative decoding. The x-axis shows the SNR of the head packet. As an example illustration, the SNR of the tail packet is set to 3dB lower than that of the head packet. *no-collision* indicates the decoding performance when only the head packet presents.

between 0 and 15 dB. For each SNR value, we simulate 5×10^4 collisions, each consisting of three copies of a randomly-generated packet of length 1024B. The results trivially extend to general cases with an arbitrary number of packets and varying size. We focus on the head and tail packets since these two adopt iterative decoding while others use hard-decoding.

Fig. 11 illustrates the BER and PER of Chorus's iterative decoding algorithm. We observe close performance between Chorus's collision resolution and the case without any collision. This implies that the BER and PER degradation caused by error propagation is negligible under practical settings.

The SNR-weighted combination of decoded packets reduces BER at the low SNR region. However, at the high SNR region, it results in lower performance than selective combination, *i.e.*, assigning weight 1 to the packet with higher SNR, and 0 to the other. This is because as SNR increases, the error propagation effect dominates the additional diversity from weighted combination. An additional observation is that our analysis of error propagation (Sec. V) matches well with the symbol-level simulation. Therefore, it can be used as the packet reception model in network-level simulation of Chorus.

An additional observation from Fig. 11 is the impact of SNR on Chorus's performance. Inaccurate channel estimation reduces the SNR, thus increasing BER. Our previous analysis assumed accurate channel estimation during the iterative decoding. This is because Chorus detects and decodes collided packets with relatively high SNR, while treating undetectable packets as noise. In addition, channel estimation is usually realized via adaptive filtering [9], thus the noise added is much lower than ambient noise and interference.

B. Network-Level Performance

We now evaluate the broadcast performance of Chorus. We implement the cognitive sensing and broadcast scheduling protocols based on the 802.11b module in ns-2. We adopt the collision-resolution module as the PHY-layer packet reception model. This module computes the SNR for a given collision pattern, following the analysis in Sec. V. The resultant SNR is then compared with the SNR threshold to determine whether the reception succeeds. We do not consider error propagation since it has negligible effect on PER, as shown in our previous analysis and simulation. We only use the selective combination when multi-packet collision occurs.

We use a typical CSMA/CA-based protocol, *Double-Coverage Broadcast* (DCB) [4] as a performance benchmark. In order to reduce the latency caused by redundant transmissions, DCB prunes the network topology, such that only those nodes

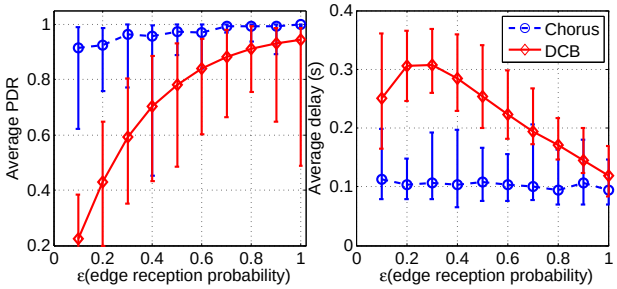

Fig. 12. The impact of link quality (reflected by ϵ) on latency and PDR. The error bars indicate variation over 30 random topologies.

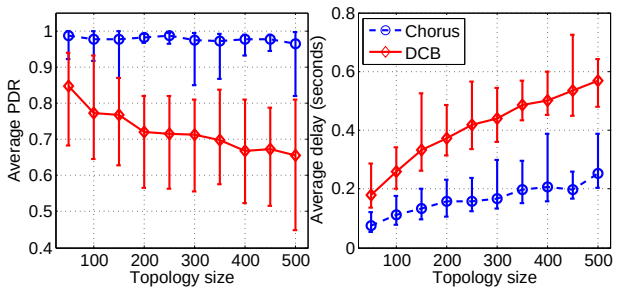

Fig. 13. Scalability of the broadcast protocols as the topology size (number of nodes) grows.

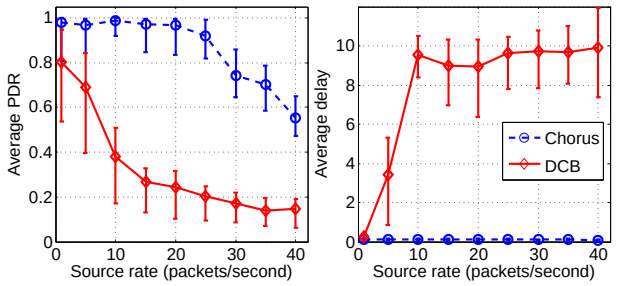

Fig. 14. Sensitivity to source rate, which indicates the maximum supportable throughput of a broadcast protocol.

with the potential to deliver packets to many downstream receivers will be selected. It further improves PDR by ensuring that each receiver is covered at least twice by other selected forwarders. DCB has been compared with a number of other CSMA/CA-based broadcast protocols and demonstrated superior performance.

We have implemented DCB based on the ns-2 802.11b MAC, following the specification of Algorithm 5 in [4]. Since it requires a strict definition of neighborhood, DCB assumes a transmission range exists, within which all nodes receive packets from the transmitter with the same probability. To improve accuracy while satisfying this requirement, we use the following channel model. We define transmission range at a distance where reception succeeds with an *edge reception probability* ϵ. Within this range, the RSS follows the log-normal distribution [15], with mean 4 and std 5 (dB). This channel model represents a middle ground between the UDG and the log-normal shadowing model. When ϵ is close to 1, it approaches the UDG model. As ϵ approaches 0, it is equivalent to a shadowing model. For a given topology, as ϵ decreases, the average link quality decreases. From the symbol-level simulation in Fig. 11, we observe a sharp decrease of PER beyond certain SNR. Therefore, it is reasonable to assume a SNR threshold exists, above which packets cannot be received. Given the edge reception probability ϵ and noise power, the SNR threshold is calculated by inverting the log-normal function [15].

All experiments are repeated on 30 randomly-generated topologies with node degree ranging from 2 to 9. We measure PDR according to the fraction of nodes that successfully receive a packet, and latency the duration between its release and the last successful reception. Both the PDR and latency are averaged over 1000 packets for each topology, and evaluated with respect to: link quality (indicated by ϵ), network size, source rate and packet size. The typical settings are: source rate 1 pkt/s (packets/second), packet size 1KB, edge reception probability $\epsilon = 0.5$, network size (number of nodes) 100 with average node density 6. Unless noted otherwise, we isolate the effect of each factor by varying it while fixing others to the typical values.

Our experimental results on DCB are consistent with [4] at a high link quality, low source rate, small packet size and small network size. However, in the general case, DCB's performance degrades fast. In contrast, `Chorus` demonstrates significant advantages in all cases. We report the detailed experiments below.

1) Link quality: We vary the link quality by tuning the edge reception probability ϵ. A higher ϵ value implies a lower packet loss rate for average links in the network. As shown in Fig. 12, the PDR of both `Chorus` and DCB decreases with loss rate. However, `Chorus` is much less sensitive to the link condition, owing to the diversity provided by collision resolution. As ϵ varies, `Chorus`'s latency remains around 0.1 second, while DCB's latency varies from 0.12 to 0.3. More importantly, `Chorus` keeps more than 90% PDR under all link conditions, while DCB's average PDR drops from 90% to 20% as ϵ decreases. Note that DCB's latency may drop as the link quality decreases. This is at the expense of severe packet losses as indicated by the decrease of PDR.

2) Network size: Sensitivity to network size indicates the scalability of the broadcast protocol. To quantify scalability of `Chorus`, we keep the average network density to 6 while increasing the total number of nodes in the network. The network radius grows accordingly. Fig. 13 plots the resulting latency and PDR. `Chorus` demonstrates negligible loss of PDR as the networks size grows. In addition, its latency is 75% lower than that of DCB. Consistent with the asymptotic analysis, its latency increases with the network size. However, the growth rate or sensitivity to network size is much lower than DCB.

3) Source rate: It is well-known that in end-to-end unicast or broadcast, the throughput drops when the source rate is too high and the network becomes congested. Therefore, the maximum supportable source rate reflects the maximum throughput of a broadcast protocol. In Fig. 14, we vary the rate at which the source node generates broadcast packets, and track the resulting latency and PDR. Both `Chorus` and DCB's PDR decreases abruptly beyond certain margins, which roughly indicate their supportable throughput. The supportable throughput of `Chorus` is around 20 pkts/second, in contrast to 1 pkt/second in DCB. In addition, DCB' latency increases from 0.1 second to 10 seconds as the source rate increases from 1 to 40 pkts/second, while `Chorus` maintains around 0.1 second latency across this range.

4) Packet size: Fig. 15 shows how packet size affects the broadcast performance when coupled with variation of source

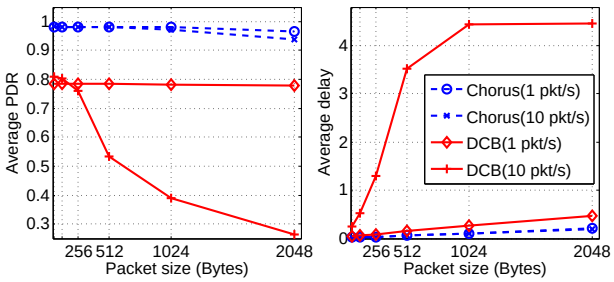

Fig. 15. Impact of packet sizes, which range from 64 to 2048 bytes.

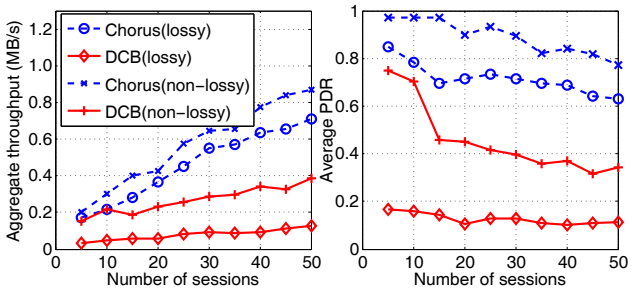

Fig. 16. Total broadcast throughput and average PDR when multiple sources transmit different data, for lossy (edge reception probability $\epsilon = 0.1$, average link quality $q = 0.51$) and non-lossy ($\epsilon = 0.5, q = 0.83$) networks.

rate. When source rate is low (1 pkt/s), the network is less congested, thus Chorus's spatial reuse advantage is less obvious. Owing to the diversity gain, however, it maintains a PDR higher than 95%, in contrast with 80% when running DCB. In addition, its latency is 60% lower than DCB for all packet sizes. When source rate is high (10 pkt/s), Chorus's PDR and latency remains the same. In contrast, DCB suffers from a sharp degradation of performance—its latency increases from 0.2 to 4 seconds as packet size grows from 64B to 1024B. Again, this is due to its limited supportable throughput. For larger packets, the source injects more data into the network per unit time, which causes congestion. In addition, the cost of losing one packet increases, resulting in higher latency and lower PDR.

As indicated in Sec. V, the worst-case delay of Chorus is affected by its packet overhead. The experiment results in Fig. 15 show that Chorus is relatively insensitive to packet overhead, in contrast to the analysis. This is because the worst case in Fig. 10 rarely occurs in a random network, and the overhead is negligible compared with packet length.

5) Multiple broadcast sessions: We proceed to evaluate the case where multiple broadcast sessions co-exist, each corresponding to one randomly selected source node in a 50-node topology. We set $\epsilon = 0.1$ and $\epsilon = 0.5$ to represent a lossy and non-lossy network, respectively. The former case is close to a real world mesh network [16] in which most links have intermediate reception rate. We focus on two metrics: average PDR among all sessions, and broadcast throughput, which equals the total amount of data delivered to all nodes within unit time, summed over all the sessions. Fig. 16 plots these metrics as a function of traffic load (the number of sessions). In a lossy network, Chorus achieves 3x higher throughput than DCB, and maintains a PDR above 60%, which indicates the friendliness among different traffic. The performance gain over DCB is less in a non-lossy network, where Chorus benefits more from spatial reuse than diversity gain. Also note that although throughput increases when the traffic load is high, the cost is lower PDR, implying that most traffic is confined to around the source nodes, especially for the DCB protocol.

VII. Conclusion

In this paper, we provide theoretical and practical results that demonstrate the feasibility and advantage of a collision-resolution protocol for wireless broadcast. We introduce Chorus, which allows forwarders with the same outgoing packets to transmit at roughly the same time, and then employs physical-layer iterative decoding to resolve collisions at the receiver. By decoding multiple versions of a packet at once, Chorus achieves transmit diversity and improves loss resilience without any retransmission. More importantly, with its collision-tolerant MAC, Chorus significantly simplifies the CSMA scheduling and improves its spatial reuse. Our theoretical analysis and symbol-level simulation show that Chorus's iterative decoding algorithm can effectively resolve collisions with negligible error propagation effect. We also establish an asymptotic latency bound of $\Theta(r)$ when using Chorus for broadcast, where r is the network radius. Our network-level experiments further show that Chorus outperforms a typical CSMA/CA-based broadcast protocol by a significant margin, in terms of latency, reliability, throughput, and scalability. These features make Chorus suitable especially for fast information dissemination in large-scale networks, such as wireless mesh networks.

References

[1] R. Gandhi, S. Parthasarathy, and A. Mishra, "Minimizing Broadcast Latency and Redundancy in Ad Hoc Networks," in *Proc. of ACM MobiHoc*, 2003.

[2] S.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang, "Minimum-Latency Broadcast Scheduling in Wireless Ad Hoc Networks," in *Proc. of IEEE INFOCOM*, 2007.

[3] S. Huang, P. J. Wan, J. Deng, and Y. Han, "Broadcast Scheduling in Interference Environment," *IEEE Trans. on Mobile Computing*, vol. 7, no. 11, Nov 2008.

[4] W. Lou and J. Wu, "Toward Broadcast Reliability in Mobile Ad Hoc Networks with Double Coverage," *IEEE Trans. on Mobile Computing*, vol. 6, no. 2, 2007.

[5] R. Mudumbai, D.R. Brown, U. Madhow, and H.V. Poor, "Distributed Transmit Beamforming: Challenges and Recent Progress," *IEEE Communications Magazine*, vol. 47, no. 2, 2009.

[6] B. S. Chlebus, Gasieniec L., A. Gibbons, A. Pelc, and W. Rytter, "Deterministic Broadcasting in Unknown Radio Networks," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[7] R. Mahjourian, F. Chen, R. Tiwari, M. Thai, H. Zhai, and Y. Fang, "An Approximation Algorithm for Conflict-Aware Broadcast Scheduling in Wireless Ad Hoc Networks," in *Proc. of ACM MobiCom*, 2008.

[8] D. Halperin, T. Anderson, and D. Wetherall, "Taking the Sting Out of Carrier Sense: Interference Cancellation for Wireless LANs," in *Proc. of ACM MobiCom*, 2008.

[9] S. Gollakota and D. Katabi, "ZigZag Decoding: Combating Hidden Terminals in Wireless Networks," in *Proc. of ACM SIGCOMM*, 2008.

[10] A. Scaglione and Y.-W. Hong, "Opportunistic Large Arrays: Cooperative Transmission in Wireless Multihop Ad Hoc Networks to Reach Far Distances," *IEEE Trans. on Signal Processing*, vol. 51, no. 8, 2003.

[11] X. Zhang and K. G. Shin, "DAC: Distributed Asynchronous Cooperation for Wireless Relay Networks," in *Proc. of IEEE INFOCOM*, 2010.

[12] B. Sklar, *Digital Communications: Fundamentals and Applications*, Prentice Hall, 2001.

[13] IEEE Standard, "802.11™: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications ," 2007.

[14] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[15] J. Camp, J. Robinson, C. Steger, and E. Knightly, "Measurement Driven Deployment of a Two-tier Urban Mesh Access Network," in *Proc. of ACM MobiSys*, 2006.

[16] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network," in *Proc. of ACM MobiCom*, 2005.