CEDAR: A Low-Latency and Distributed Strategy for Packet Recovery in Wireless Networks

Chenxi Qiu, Haiying Shen, *Senior Member, IEEE, Member, ACM*, Sohraab Soltani, Karan Sapra, Hao Jiang, and Jason O. Hallstrom, *Member, IEEE*

***Abstract*—Underlying link-layer protocols of well-established wireless networks that use the conventional "store-and-forward" design paradigm cannot provide highly sustainable reliability and stability in wireless communication, which introduce significant barriers and setbacks in scalability and deployments of wireless networks. In this paper, we propose a Code Embedded Distributed Adaptive and Reliable (CEDAR) link-layer framework that targets low latency and balancing en/decoding load among nodes. CEDAR is the first comprehensive theoretical framework for analyzing and designing distributed and adaptive error recovery for wireless networks. It employs a theoretically sound framework for embedding channel codes in each packet and performs the error correcting process in selected intermediate nodes in a packet's route. To identify the intermediate nodes for the decoding, we mathematically calculate the average packet delay and formalize the problem as a nonlinear integer programming problem. By minimizing the delays, we derive three propositions that: 1) can identify the intermediate nodes that minimize the propagation and transmission delay of a packet; and 2) and 3) can identify the intermediate nodes that simultaneously minimize the queuing delay and maximize the fairness of en/decoding load of all the nodes. Guided by the propositions, we then propose a scalable and distributed scheme in CEDAR to choose the intermediate en/decoding nodes in a route to achieve its objective. The results from real-world testbed "NESTbed" and simulation with MATLAB prove that CEDAR is superior to schemes using hop-by-hop decoding and destination decoding not only in packet delay and throughput but also in energy-consumption and load distribution balance.**

***Index Terms*—Link-layer protocol, low-latency, reliability and stability, wireless networks.**

I. Introduction

DESPITE the unprecedented success and proliferation of wireless communication, there are major shortcomings in the underlying link-layer protocols in providing sustainable reliability and stability among wireless users. Popular wireless link-layer protocols, such as the retransmission ARQ or forward error correction (FEC)-based ARQ (HARQ) approaches (employed by the IEEE 802.xx and LTE standard suite) are designed to achieve some level of reliability by discarding a corrupted packet at the receiver and performing one or more retransmissions until the packet is decoded/received error-free or a maximum number of retransmission attempts is reached. This methodology suffers from degradation of throughput and overall system instability since decoding failures at the receiver due to a small number of bit errors lead to packet drops and discarding a large number of correctly delivered data bits.

Many leading research efforts [1]–[12] have highlighted the inefficiencies of these link-layer protocols and proposed a variety of remedy solutions. The majority of these efforts either consider variations of the ARQ, HARQ, or a hybrid approach of both schemes [1], [5], [9], [11]–[13]. They largely follow the traditional "store-and-forward" link-layer design paradigm: Each data packet must be fully received and corrected by every relay node before it is forwarded. This design paradigm increases stability, but still cannot provide high stability due to its hop-by-hop operation.

It is our belief that achieving the ultimate objective of the development of ubiquitous and heterogeneous wireless networks demands fundamental and radical changes to the conventional link-layer protocol design. Thus, we study and develop alternative optimal and low-complexity error recovery strategies in link-layer design to achieve high reliability and stability by partially and optimally selecting relay nodes. The objectives of the strategies are to ensure: 1) low end-to-end latency and rapid delivery of packets; 2) high throughput with minimum data loss. To meet these objectives, we develop solutions that address the following key issues: 1) *minimizing propagation and transmission (prop&tran) delay*: At which intermediate nodes (if any) should a link-layer packet be detected to minimize packet delay? 2) *minimizing queuing delay*: As multiple relay nodes in a route perform error recovery on the same packet stream and one node may perform error recovery for multiple packet streams, how should we select relay nodes that provide global reliability and stability in a wireless network with many source–destination packet streams? 3) *energy-efficiency*: what is the optimal processing contribution of each relay node for error recovery with respect to reliability and energy consumption? Note that our work shares the same objectives as some previous works on en/decoding schemes and network coding (e.g., PPR [9] and MIXIT [12]). However, unlike these previous works that focus on route determination or en/decoding scheme design, our work aims to determine the intermediate nodes to en/decode packets given a route and an en/decoding scheme. Our work can be employed in those en/decoding schemes and network coding schemes for further performance enhancement.

The design of the strategies requires network-of-queues models that capture the error correction process and networking

Manuscript received January 17, 2013; revised September 05, 2013 and April 09, 2014; accepted June 18, 2014; approved by IEEE/ACM Transactions on Networking Editor L. Qiu. *(Corresponding author: Haiying Shen.)*

C. Qiu, H. Shen, and K. Sapra are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 USA (e-mail: chenxiq@clemson.edu; shenh@clemson.edu; ksapra@clemson.edu).

S. Soltani is with Intelligent Automation, Inc., Rockville, MD 20855 USA (e-mail: ssoltani@i-a-i.com).

H. Jiang and J. O. Hallstrom are with the School of Computing, Clemson University, Clemson, SC 29634 USA (e-mail: hjiang@clemson.edu; jasonoh@clemson.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2332980

effects of traffic flows over multihop wireless paths. Accordingly, we develop mathematical models for the prop&tran delay and queuing delay for a packet based on the path length between two consecutive decoding nodes in a route (route segment length). Through rigorous mathematical analysis on the models, we derive two propositions that: 1) can identify the intermediate nodes for decoding that minimize prop&tran delay of a packet; and 2) prove that balanced en/decoding load distribution among decoding nodes in the network minimizes the queuing delay. Based on the propositions, we formulate the problem of minimizing delay as a nonlinear integer programming problem. However, due to the NP-hard nature of the problem and impracticability of collecting all required information to find the global optimal solution, we propose a suboptimal Code Embedded Distributed Adaptive and Reliable (CEDAR) link-layer framework for wireless networks. CEDAR is a distributed and cooperative error recovery design, which represents a new paradigm in both transmission and distributed recovery processing and promises significant increase of capacity and throughput gain in wireless networks. CEDAR provides an adaptive environment for various error recovery strategies with respect to reliability, stability, and energy consumption constraints. We believe CEDAR is the first comprehensive theoretical framework for analyzing and designing distributed and adaptive link-layer error recovery that targets system reliability, stability in conjunction with optimal energy consumption for wireless communication.

We summarize our contributions as follows. 1) We build a model for the probability of decoding failure of a packet traveling through a given number of hops based on the finite-state Markovian channel (FSMC) model. 2) We build rigid mathematical models for the prop&tran delay, and queuing delay for a packet, and en/decoding load of each node. 3) We formalize the problem of choosing the intermediate en/decoding nodes for minimum delay and minimum difference of en/decoding load of all the nodes as a nonlinear integer programming problem that is an NP-hard problem. 4) We propose a distributed suboptimal strategy for CEDAR that achieves high reliability, stability, and en/decoding load balancing.

The reminder of the paper is organized as follows. Section II states the problem that needs to be solved to minimize the packet delay. Section III introduces a mathematical model that formalizes the problem as a nonlinear integer programming problem and derives two propositions to minimize the delay. Guided by the propositions, Section IV details the design of CEDAR for solving the problem in Section II, and Section V presents performance evaluation of CEDAR in comparison to previous schemes. Section VI presents a review of the related works. The final section concludes with a summary of contributions and a discussion on future work.

II. Problem Statement

Consider a wireless network composed of N *nodes* denoted by $\mathcal{V} = \{v_1, \ldots, v_N\}$. Each traffic flow from a *source node* to a *destination node* transverses over a predetermined set of *links* (a route specified by the network layer). Let $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_K\}$ denote the set of transmission routes. Each route $\mathbf{r}_k$ $(\mathbf{r}_k \in \mathcal{R})$ carries a data stream following Poisson distribution with arrival rate λ_k. We use $\mathbf{r}_k = \{v_{k_1}, \ldots, v_{k_{n_k}}\}$ to represent the node sequence in $\mathbf{r}_k$ $(v_{k_1}, \ldots, v_{k_{n_k}} \in \mathcal{V})$, where n_k is the number of nodes in $\mathbf{r}_k$. We consider a network with heterogeneous types of traffic, i.e., a combination of real-time traffic with delay constraint and traffic with no delay constraint. We use U_k to denote the delay constraint of route r_k; $U_k = \infty$ if the packet in route r_k has no delay constraint. Finally, we use indicator variable $y_{i,k}$ to denote whether v_i is in $\mathbf{r}_k$. If yes, $y_{i,k} = 1$; otherwise $y_{i,k} = 0$.

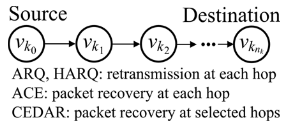

Source
Destination
v_{k_0}
v_{k_1}
v_{k_2}
$v_{k_{n_k}}$
ARQ, HARQ: retransmission at each hop
ACE: packet recovery at each hop
CEDAR: packet recovery at selected hops

Fig. 1. Protocols for packet recovery.

As shown in Fig. 1, to reach the destination, each packet flow needs to travel through all nodes in the predetermined route, and some of these nodes are responsible for en/decoding the packets. In the ARQ and HARQ protocols [1], [5], [11], each hop drops distorted packets and requests for complete or partial retransmission of the original packets. These methods follow the conventional link-layer design paradigm and guarantee the reliability between any pair of nodes. However, this strategy causes high delays and low throughput (due to numerous retransmissions at every relay hop), leading to significant degradation in channel bandwidth utilization. Furthermore, although decoding in each hop (adopted by the HARQ family) increases the reliability, it comes at the cost of high computational overhead. In recently proposed schemes (e.g., ACE [15]), each relay node stores an erroneous received packet for packet recovery (with no retransmission requirements) until the packet is corrected before forwarding it to the next hop. Though these schemes overcome the shortcomings of the ARQ and HARQ protocols to a certain extent, they are still not effective in achieving high throughput and low energy and bandwidth consumption.

CEDAR introduces a new flexible environment for link-layer error recovery: 1) it employs a theoretically sound framework and a corresponding strategy for embedding channel codes, using robust and adaptive code rates, in each packet; 2) the error correction process is performed in a distributed and optimal manner where selected (and not all) intermediate nodes participate in performing error recovery. The key problem in CEDAR is how to identify candidates among the intermediate nodes for the CEDAR en/decoding process to optimally decrease the overall delay and increase throughput and fairness of en/decoding load over the entire network.

To this end, first, we build models to calculate the delay $(D(\mathbf{n}_i))$ and the en/decoding load $(L(\mathbf{n}_i))$ of each intermediate node v_i based on the lengths of the routing paths (denoted by $\mathbf{n}_i$) of the packets crossing v_i. We use these models to calculate the expected delays and en/decoding load of each node, and ultimately identify the positions of intermediate nodes for en/decoding in each route in CEDAR. Throughout the paper, we use the key terms provided in the following definitions.

Definition 2.1 (En/Decoding load): The en/decoding load of v_i, denoted by $L(\mathbf{n}_i)$, is defined as the sum of the arrival rates for all the packet streams that v_i is responsible for en/decoding.

Definition 2.2 (Key Node): A key node of route $\mathbf{r}_k$ is a node responsible for en/decoding the packets traveling along $\mathbf{r}_k$. Matrix $\mathbf{X} = (x_{i,k})_{N \times K}$ denotes whether v_i is a key node in $\mathbf{r}_k$:

$$x_{i,k} = \begin{cases} 1, & v_i \text{ is the key node in } \mathbf{r}_k \\ 0, & v_i \text{ is not the key node in } \mathbf{r}_k. \end{cases} \tag{1}$$

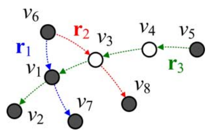


Fig. 2. Route segment.

Definition 2.3 (Route Segment): A route segment of $\mathbf{r}_k$ is a section of the end-to-end path between one key node to either the endpoints or another key node. The length of a route segment is defined as the number of hops in the route segment.

In each route segment, the packet sender (the first key node) encodes the packets, and the packet receiver (the second key node) decodes the packets. In other words, the second key node is responsible for decoding for its route segment. Use matrix $\mathbf{N} = (n_{i,k})_{N\times K}$ to denote the length of a route segment with decoding node v_i in $\mathbf{r}_k$, and use vector $\mathbf{n}_i$ denote the lengths of route segments responsible by v_i for all the routes, i.e., $\mathbf{N} = [\mathbf{n}_1,\dots,\mathbf{n}_N]^T$ and $\mathbf{n}_i = [n_{i,1},\dots,n_{i,K}]$. Here, we define $n_{i,k} = 0$ if v_i has no responsibility of decoding the packet in $\mathbf{r}_k$. For example, in Fig. 2, there are eight nodes $\mathcal{V} = \{v_1,\dots,v_8\}$, and three routes $\mathcal{R} = \{\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3\}$, where $\mathbf{r}_1 = \{v_6,v_1,v_7\}$, $\mathbf{r}_2 = \{v_6,v_3,v_8\}$, and $\mathbf{r}_3 = \{v_5,v_4,v_3,v_1,v_2\}$. v_6, v_1, and v_7 are the key nodes in $\mathbf{r}_1$; v_6, v_3, and v_8 are the key nodes in $\mathbf{r}_2$; v_5, v_1, and v_2 are the key nodes in $\mathbf{r}_3$. Then, we can derive that $n_{1,1} = 1$ and $n_{1,2} = 3$ because there are one hop from v_6 to v_1 in $\mathbf{r}_1$ and three hops from v_5 to v_1 in $\mathbf{r}_2$. Also, $n_{1,3} = 0$ because v_1 is not responsible for decoding packets in $\mathbf{r}_3$. Hence, $\mathbf{n}_1 = [n_{1,1}\ n_{1,2}\ n_{1,3}] = [1\ 3\ 0]$.

Let $\lambda_{i,k}$ denote the arrival rate of the data stream that v_i is responsible for en/decoding in $\mathbf{r}_k$. Then, $\lambda_{i,k} = \lambda_k \times x_{i,k}$. We use $\overline{D(\mathbf{n}_i)}$ to denote the average delay when a packet crosses v_i with route segment vector $\mathbf{n}_i$. Then, the total average packet delay decoding at v_i within a unit time equals $\overline{D(\mathbf{n}_i)}\sum_{k=1}^{K}\lambda_{i,k}$. Also, we use $\overline{L(\mathbf{n}_i)}$ to represent the average en/decoding load of v_i and use $\overline{L(\mathbf{N})}$ to represent the average en/decoding load of all the nodes in $\mathcal{V}$. Then, $\overline{L(\mathbf{N})} = \frac{\sum_{i=1}^{N}\overline{L(\mathbf{n}_i)}}{N}$.

Objective: The objectives of CEDAR are: 1) to minimize the total delay of the packets in the entire system, which can be represented as

$$\min \sum_{i=1}^{N}\left(\overline{D(\mathbf{n}_i)}\sum_{k=1}^{K}\lambda_{i,k}\right) \tag{2}$$

and 2) to balance the en/decoding load of all the nodes in the network. In this paper, we use standard deviation [14] of en/decoding loads, which reflects how much variation exists between each node's en/decoding load and $\overline{L(\mathbf{N})}$, to measure the balance of the en/decoding load of the network. The lower the value of the standard deviation, the higher the fairness of the en/decoding load of all the nodes. Then, the objective can be formulated as

$$\min \sqrt{\sum_{i=1}^{N}\left(\overline{L(\mathbf{n}_i)} - \overline{L(\mathbf{N})}\right)^2}. \tag{3}$$

Consequently, we combine these two objectives and formulate the optimization problem as

$$\min\ \gamma_1\sum_{i=1}^{N}\left(\overline{D(\mathbf{n}_i)}\sum_{k=1}^{K}\lambda_{i,k}\right) + \gamma_2\sqrt{\sum_{i=1}^{N}\left(\overline{L(\mathbf{n}_i)} - \overline{L(\mathbf{N})}\right)^2}$$

$$\text{s.t. } x_{i,k} \le y_{i,k},\ \overline{D(\mathbf{n}_i)} \le U_k,\ 1\le i\le N, 1\le k\le K \tag{4}$$

where γ_1 and γ_2 represent the weights we set for these two objectives. In this paper, we primarily consider minimizing packet delay and secondarily consider balancing en/decoding load. Thus, we set $\gamma_1 \gg \gamma_2$ in our system.

Now, we need to consider how to solve the multiple objective optimization problem: The packet delay in v_i (which is composed of prop&tran and queueing delays) is a function of $\mathbf{n}_i$. This will be deduced in the mathematical analysis in Section III. We use $\overline{D_{\text{p\&t}}(\mathbf{n}_i)}$ to denote the average prop&tran delay of all the packet streams being decoded in v_i, and use $\overline{D_{\text{q}}(\mathbf{n}_i)}$ to denote the average queuing delay of the packet stream in v_i. Then, the total average delay when one packet crosses v_i is

$$\overline{D(\mathbf{n}_i)} = \overline{D_{\text{q}}(\mathbf{n}_i)} + \overline{D_{\text{p\&t}}(\mathbf{n}_i)}. \tag{5}$$

Thus, we need to minimize $\overline{D_{\text{q}}(\mathbf{n}_i)}$ and $\overline{D_{\text{p\&t}}(\mathbf{n}_i)}$ in order to achieve the objective of CEDAR in (4). To this end, in Sections III-A and III-B, we model the bit error rate (BER) fluctuations of wireless channels and probability of successful decoding. Sections III-C and III-D use this model to formulate the prop&tran delay $\overline{D_{\text{p\&t}}(\mathbf{n}_i)}$ and the queuing delay $\overline{D_{\text{q}}(\mathbf{n}_i)}$. Finally, Section III-E derives two propositions to minimize $\overline{D_{\text{q}}(\mathbf{n}_i)}$ and $\overline{D_{\text{p\&t}}(\mathbf{n}_i)}$, respectively. Guided by the propositions, we design CEDAR in Section IV.

III. Mathematical Modeling

In this section, we first present a Markovian wireless channel model to capture the variations in wireless error conditions due to nonstationary wireless noise and calculate BER of a packet when it goes through several channels. Using this model, we analyze the relationship between the number of hops a packet crosses and the probability of its successful decoding. This relationship leads us to calculate the prop&tran delay and queuing delay, respectively. By minimizing the two delays, we can find the locations of intermediate nodes in a route for decoding. Finally, we formulate the problem of minimizing the sum of the delays as a nonlinear integer programming problem. The analytical results and the formed problem lay the foundation for the design of an optimized strategy for choosing intermediate nodes for the CEDAR packet recovery.

A. Markovian Channel Model

FSMC model [15] is a channel model that uses finite-state Markov chain to describe the process, under which errors are introduced into a transmitted packet over a wireless route. The model has a finite set of error states $\mathcal{S} = (s_1,s_2,\dots,s_B)$ ($|\mathcal{S}| = B$), each corresponding to a binary symmetric channel (BSC). The channel model can be considered as a combination of B number of various BSCs with unique BERs (ϵ) (i.e., $\epsilon_l \ne \epsilon_j$ for $l\ne j$, $l,j = 1,2,\dots,B$). Assume packets are transmitted during discrete time-slots τ_i ($i = 1,2,3\dots$) that can be referred to as *transmission intervals*. During the ith transmission interval, a packet is transmitted from a BSC to another BSC with crossover BER ϵ_i. Each ϵ_i of a particular τ_i is valued from $\mathcal{S}$. The Markovian model assumes a homogenous and stationary Markov chain with transition probability matrix $\mathcal{T} = (t_{ij})_{B\times B}$ and initial probability $\boldsymbol{\pi} = (\pi_1,\dots,\pi_B)$. $\mathcal{T} = (t_{ij})_{B\times B}$ can be trained on real channel traces by using the statistics of previous transmission intervals. This captures the effects of multipath fading and interferences on the channel BER in every transmission interval using a single aggregated model [15]. The system average BER can be calculated as: $\bar{\epsilon} = \sum_{k=1}^{B} \pi_k \epsilon_k$. Based on this prior work, we calculate the average BER for consecutive wireless links within a route segment in a cascaded system, and derive Lemma 3.1.

Lemma 3.1: The BER in a cascade system where a node travels along links with states $s_{a_1} s_{a_2} \dots s_{a_n}$ $(1 \le a_1, a_2, \dots, a_n \le K)$ can be given by

$$\overline{\epsilon^n} \approx \sum_{\{s_{a_1} \dots s_{a_n}\} \in \mathcal{S}^n} \left(\pi_{a_1} \prod_{j=1}^{n-1} t_{a_j a_{j+1}} \sum_{i=1}^{n} \epsilon_i \right) \tag{6}$$

where $\mathcal{S}^n$ represents all the possible sets of series that are composed of n elements and each element is contained in S (notice each series can have duplicated elements).

Proof: Let $\mathcal{E}_i = \begin{bmatrix} 1-\epsilon_i & \epsilon_i \\ \epsilon_i & 1-\epsilon_i \end{bmatrix}$ be the transition probability matrix when the packet's channel is in s_i. We then derive that

$$\mathcal{E}_i = \mathcal{B}^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1-2\epsilon_i \end{bmatrix} \mathcal{B} \tag{7}$$

where $\mathcal{B} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Then, we consider the situation that one bit goes through the cascade of n nodes and the bit's channel state is changed in the sequence of $s_{a_1}, s_{a_2}, \dots s_{a_i}, \dots, s_{a_n}$ $(1 \le a_i \le K, 1 \le i \le n)$. In this case, the transition probability matrix through n nodes, denoted as $\mathcal{E}_{a_1 a_2 \dots a_n}$, is given by

$$\begin{aligned} \mathcal{E}_{a_1 a_2 \dots a_n} &= \mathcal{E}_{a_1} \mathcal{E}_{a_2} \dots \mathcal{E}_{a_n} = \mathcal{B}^{-1} \begin{bmatrix} 1 & 0 \\ 0 & \prod_{i=1}^{n} (1-2\epsilon_{a_i}) \end{bmatrix} \mathcal{B} \\ &= \begin{bmatrix} \frac{1+\prod_{i=1}^{n}(1-2\epsilon_{a_i})}{2} & \frac{1-\prod_{i=1}^{n}(1-2\epsilon_{a_i})}{2} \\ \frac{1-\prod_{i=1}^{n}(1-2\epsilon_{a_i})}{2} & \frac{1+\prod_{i=1}^{n}(1-2\epsilon_{a_i})}{2} \end{bmatrix}. \end{aligned}$$

Thus, the BER of the cascade of n nodes $(a_1, a_2, a_3, \dots, a_n)$ equals

$$\epsilon_{a_1 a_2 \dots a_n} = \frac{1 - \prod_{i=1}^{n} (1-2\epsilon_{a_i})}{2}. \tag{8}$$

The probability that such an aforementioned situation occurs equals

$$\Pr[X = \{s_{a_1} s_{a_2} \dots s_{a_n}\}] = \pi_{a_1} \prod_{j=1}^{n-1} t_{a_j a_{j+1}} \tag{9}$$

where X is a random variable that represents the series. Then, the expectation of error bit through the cascade of n hops is given by

$$\begin{aligned} \overline{\epsilon^n} &= \sum_{\{s_{a_1} \dots s_{a_n}\} \in \mathcal{S}^n} \Pr[X = \{s_{a_1} s_{a_2} \dots s_{a_n}\}] \times \epsilon_{a_1 a_2 \dots a_n} \\ &= \sum_{\{s_{a_1} \dots s_{a_n}\} \in \mathcal{S}^n} \pi_{a_1} \prod_{j=1}^{n-1} t_{a_j a_{j+1}} \frac{1 - \prod_{i=1}^{n} (1-2\epsilon_{a_i})}{2}. \end{aligned} \tag{10}$$

When $\epsilon_1, \epsilon_2, \dots, \epsilon_n \ll 1$

$$\overline{\epsilon^n} \approx \sum_{\{s_{a_1} \dots s_{a_n}\} \in \mathcal{S}^n} \left(\pi_{a_1} \prod_{j=1}^{n-1} t_{a_j a_{j+1}} \sum_{i=1}^{n} \epsilon_{a_i} \right). \tag{11}$$

□

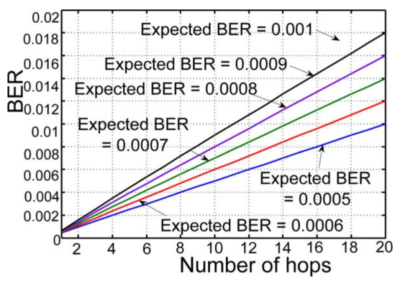


Fig. 3. Curve of (6).

B. Probability of Successful Decoding

CEDAR is developed based on the error recovery mechanism in the ACE Communication Model [16]. Thus, we first introduce ACE before we present the mathematical models. Specifically, during τ_i, a transmitter encodes data symbols z_i with parity codes x_i (referred to as type-I parity code) to create a codeword $C_i(z_i, x_i)$. It transmits a packet $M_i = (C_i(z_i, x_i), y_i)$, where y_i denotes the additional parity (hereafter type-II parity) symbols for recovering previously received corrupted packets at the receiver. We also use x_i, y_i, and z_i to denote the number of their symbols. The receiver utilizes x_i to decode C_i. If the decoding operation fails, the receiver stores C_i in its buffer and issues a request along with ACK_i for more parity symbols. The transmitter then sends additional parity $y_j (j > i)$ along with M_j. We use $m_i = x_i + y_i$ to denote the total number of parity symbols of M_i.

First, consider a simple cascade model $(v_0 \to v_1 \to \dots \to v_n)$ in which a packet stream goes through a series of nodes $v_0, v_1, v_2, \dots, v_n$ and is encoded and decoded at v_0 and v_n, respectively. Fig. 3 shows an example of the curve of (6), where n is varied from 1 to 10, and the expected value of $\bar{\epsilon}$ is varied from 0.0005 to 0.001. We assume the channel has two states—noisy and not noisy—and each state can transfer to the other state at the next time-slot with the same probability. From the figure, we can find that $\overline{\epsilon^n}$ is approximately proportional to n. Thus, we drew a figure based on (6) and found that $\overline{\epsilon^n}$ is approximately proportional to n. We can approximate (6) by (12) to calculate the BER for a routing through n nodes under the FSMC

$$\overline{\epsilon^n} \approx n\bar{\epsilon}. \tag{12}$$

As ACE, we take Reed–Solomon codes [17] as an example, which is a kind of nonbinary cyclic error-correcting codes, for channel coding. In the Reed–Solomon codes, each symbol is composed of b bits, indicating that the probability of error for each symbol equals: $\overline{\epsilon^{n,b}} = 1 - \left(1 - \overline{\epsilon^n}\right)^b$. The number of error symbols introduced in one packet M_i with a length of $z_i + m_i$ symbols through n hops can be represented by a random variable E_i following a binomial distribution $E_i \approx Bi\left(z_i + m_i, \epsilon^{n,b}\right)$. If the error estimate is $\hat{\epsilon}^{n,b}$ for one symbol of b bits, the receiver is capable of correcting up to αm_i errors out of $|C_i|$ symbols in packet M_i, where α is a function measuring the expected error-correcting capability of a particular decoder based on $\hat{\epsilon}^{n,b}$. For instance, the error-correcting capability of the Reed–Solomon codes is half as many as redundant symbols (i.e., $\alpha = 0.5$) [17]. The probability

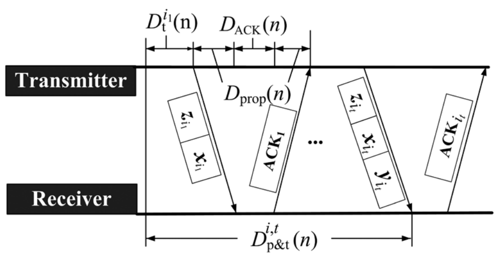


Fig. 4. Transmission and propagation delay.

of successfully recovering data bits by a parity code with m_i length symbols equals

$$\Pr\left[E_i \le \alpha m_i\right] = \sum_{k=0}^{\lfloor \alpha m_i \rfloor} \binom{z_i + m_i}{k} \left(1 - \hat{\epsilon}^{n,b}\right)^{z_i + m_i - k} \left(\hat{\epsilon}^{n,b}\right)^k. \tag{13}$$

From (13), we observe that the probability of successful is a discrete function of two variables

$$\Pr\left[E_i \le \alpha m_i\right] = F_\alpha\left(n, m_i\right). \tag{14}$$

$F_\alpha\left(n, m_i\right)$ is monotonically decreasing function of n (number of hops in a route) and is a monotonically increasing function of m_i (number of symbols in parity code).

Based on (14), the number of times (i.e., trials) a packet is required to be decoded until it is recovered has a nonhomogeneous geometric distribution (denoted by G) [18] given that the length (i.e., number of symbols) of predetermined parity code equals m_t at the tth trial.

Lemma 3.2: We use $f_G^t\left(n, m_{i_t}\right)$ to denote the probability of successful decoding on the tth decoding trial for a packet M_i going through n hops. Then

$$f_G^t\left(n, m_{i_t}\right) = F_\alpha\left(n, m_{i_t}\right) \times \prod_{j=1}^{t-1}\left(1 - F_\alpha\left(n, m_{i_j}\right)\right). \tag{15}$$

C. Propagation and Transmission Delay

In this section, we consider the prop&tran delay of each packet M_i. We use $D_{\mathrm{p\&t}}^{i,t}\left(n\right)$ to denote the prop&tran delay of M_i when the parity code of M_i has been transmitted for t times through n nodes. Let $D_{\mathrm{p}}\left(n\right)$ represent the propagation delay for one packet going through n nodes and $D_{\mathrm{ACK}}\left(n\right)$ denote the transmission delay of the ACK packet. Furthermore, let $D_{\mathrm{t}}^{i_k}\left(n\right)$ denote the transmission delay of the packet M_{i_k}. The length of this packet is $L_{\mathrm{pac}}^{i_k} = z_{i_k} + m_{i_k}$, where M_{i_k} is the kth packet that carries M_i's parity symbols for the kth time after $k-1$ times of recovery failures (i.e., type-II parities). Then, as Fig. 4 shows, $D_{\mathrm{p\&t}}^{i,t}\left(n\right)$ can be calculated as

$$D_{\mathrm{p\&t}}^{i,t}\left(n\right) = \sum_{k=0}^{t-1}\left(2D_{\mathrm{p}}\left(n\right) + D_{\mathrm{ACK}}\left(n\right) + D_{\mathrm{t}}^{i_k}\left(n\right)\right) + D_{\mathrm{p}}\left(n\right) + D_{\mathrm{t}}^{i_t}\left(n\right). \tag{16}$$

We use $R_{i,l}$ to denote the bandwidth provided to the route M_i travels in the lth hop. Assume electric signal travels at velocity c in the media and the distance of each hop (d) is an invariable. Then, $D_{\mathrm{ACK}}\left(n\right)$, $D_{\mathrm{t}}^{i_k}\left(n\right)$, and $D_{\mathrm{p}}\left(n\right)$ can be calculated as

$$D_{\mathrm{ACK}}(n) = \frac{nL_{\mathrm{ACK}}}{R_{i,l}} \quad D_{\mathrm{t}}^{i_k}\left(n\right) = \frac{nL_{\mathrm{pac}}^{i_l}}{R_{i,l}} \quad D_{\mathrm{p}}\left(n\right) = \frac{nd}{c}. \tag{17}$$

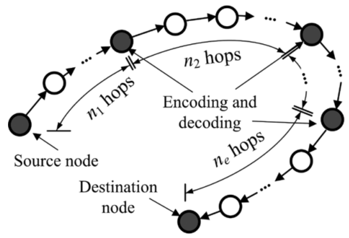


Fig. 5. Route segment.

Based on (16) and (17), we can derive that

$$D_{\mathrm{p\&t}}^{i,t}\left(n\right) = \sum_{l=1}^{n}\left[\sum_{k=0}^{t-1}\left(\frac{2d}{c} + \frac{L_{\mathrm{ACK}} + L_{\mathrm{pac}}^{i_k}}{R_{i,l}}\right) + \frac{d}{c} + \frac{L_{\mathrm{pac}}^{t}}{R_{i,l}}\right]. \tag{18}$$

Based on (15) in Lemma 3.2 and (18), we retrieve Lemma 3.3 for the expectation of $D_{\mathrm{p\&t}}^{i}\left(n\right)$.

Lemma 3.3: Assuming each packet has the same length, the expected propagation and transmission delay of a packet M_i $D_{\mathrm{p\&t}}^{i}\left(n\right)$ can be calculated by

$$\overline{D_{\mathrm{p\&t}}\left(n\right)} = \sum_{\mathrm{t}=1}^{\infty} f_G^t\left(n, m_{i_{\mathrm{t}}}\right) D_{\mathrm{p\&t}}^{i,t}\left(n\right). \tag{19}$$

As shown in Fig. 5, given a route from a source node to a destination node, we can divide the route into e segments, each segment having length of $n_1, n_2, \ldots, n_e$. In each route segment, a packet is encoded at the first node and decoded at the last node. The goal of our scheme is to determine the $n_1, n_2, \ldots, n_e$ in order to minimize the prop&tran delay of a packet from the source to the destination, i.e., to achieve

$$\min \sum_{j=1}^{e} \overline{D_{\mathrm{p\&t}}\left(n_j\right)}$$
$$\text{s.t.} \sum_{j=1}^{e} n_j = n.$$

Note that the above formulated optimization problem still holds with routing delay constraint since minimizing $\overline{D_{\mathrm{p\&t}}\left(n_j\right)}$ is sufficient for satisfying the delay constraint. That is, if the minimum value of $\overline{D_{\mathrm{p\&t}}\left(n_j\right)}$ is no larger than the constraint, then the constraint is always satisfied; otherwise, there is no solution to satisfy the constraint.

Error Estimation Code: Recall that in the above method, a receiver requests its sender to send the packet repetitively until it can successfully decode the packet, which may lead to multiple retransmissions for each packet. In order to avoid such retransmissions, we introduce another method that only needs one retransmission.

In this method, after receiving a packet, each receiver first uses error estimation code (EEC) [19] to estimate the number of corrupted symbols in the received packet, and then sends a request to its sender to ask for the additional parity code, which helps successfully recover the packet.

EEC estimates BER (e.g., checks whether BER is no larger than 1%) of the received packet, but in CEDAR, the receiver needs to estimate the number of corrupted symbols. Then, CEDAR uniformly samples the symbols instead of bits and builds EEC for the sampled symbols. More specifically, for a packet with length m_i, there are $\lfloor \log_2 m_i \rfloor$ levels of EEC bits added in each packet, with s EEC bits in each level. An EEC bit at level i $(1 \leq i \leq \lfloor \log_2 m_i \rfloor)$ is simply the parity bit for $2^i - 1$ randomly chosen symbols in the packet, which has totally $(2^i - 1)b$ bits. Each of these $2^i - 1$ data symbols is chosen uniformly randomly and independently (with replacement) from the original m_i symbols.

In addition to the parity code, each packet M_i also contains EEC codes, which has a length of $\log_2 \lfloor m_i \rfloor s/b$. Note that though this EEC-based method reduces the number of retransmission, it increases the transmission packet size. First, we consider an ideal scenario, in which EEC never underestimates the number of corrupted symbols for each packet. Then, the prop&tran delay of a packet M_i can be simply calculated by

$$D^{\text{id}}_{\text{p\&t}}(n) = D^{i,1}_{\text{p\&t}}(n) + D^{i,2}_{\text{p\&t}}(n). \tag{20}$$

However, like any error estimator, EEC may underestimate the number of errors. We use η to denote the probability of underestimation, then the actual expected prop&tran delay of a packet M_i is given by

$$\overline{D_{\text{p\&t}}(n)} = (1-\eta) D^{\text{id}}_{\text{p\&t}}(n) + \eta \sum_{\text{t}=3}^{\infty} h^{3,t}_G\left(n, m'_{i_t}\right) \sum_{l=1}^{t} D^{i,l}_{\text{p\&t}}(n) \tag{21}$$

where $h^{3,t}_G\left(n, m'_{i_t}\right) = F_\alpha\left(n, m'_{i_3}\right) \prod_{j=3}^{t-1}\left(1 - F_\alpha\left(n, m'_{i_j}\right)\right)$ and $m'_{i_j} = m_{i_j} + \log_2 \lfloor m_{i_j} \rfloor s/b$.

D. Queuing Delay

In a priority queuing model, packets entering a buffer are classified into several different priority categories and added into different queues accordingly. The packets with lower priority can enter the server only when all queues for higher-priority queues are empty. In the wireless network, for any single node v_i that is responsible for decoding at most K routes, there will be K Poisson streams $(\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,K})$ arriving at this node. Notice if v_i is not responsible for decoding packet for $\mathbf{r}_k$, $\lambda_{i,k} = 0$. v_i needs to decide the order of arriving packets to decode. Thus, by regarding v_i as the server in the model, we can use the priority queuing model (M/M/1/∞/∞/PR) [20] for analyzing the queuing delay. Note that when a packet fails to decode, it will be decoded (i.e., join in a queue) again when it received another type-II parity code along with another packet. In order to balance the queuing delay of each node, we propose a strategy for determining the priority of decoding packets. That is, the more times a packet has failed to be corrected, the higher priority it will be given when it is redecoded. When a packet suffers P number of failures, it is dropped. We do not consider the stream of retransmission for packets after P failures because the probability of failing more than P times is extremely small.

Poisson process is widely used to describe the data traffic in wireless networks [20]–[23], so we also use Poisson process to model the data traffic in this paper. The self-similar model has been proven to be more realistic than Poisson process to describe data traffic in modern LANs and WANs, in which batch arrivals, event correlations, and traffic burstiness are key factors [24]. To the best of our knowledge, there is no previous work that has studied the priority queuing system based on the self-similar model. We will use the self-similar model to analyze the packet delay in our future work.

We use priority queuing model to analyze the queuing delay for the packets crossing a given node. Fig. 6 gives a sketch of the priority queuing model in our scheme. In the figure,

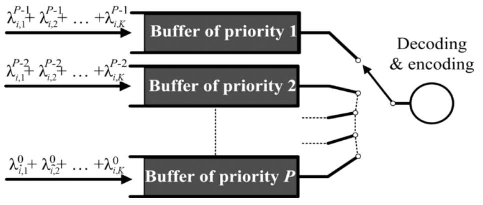

$\lambda^{P-1}_{i,1}+\lambda^{P-1}_{i,2}+\ldots+\lambda^{P-1}_{i,K}$
Buffer of priority 1
Decoding & encoding
$\lambda^{P-2}_{i,1}+\lambda^{P-2}_{i,2}+\ldots+\lambda^{P-2}_{i,K}$
Buffer of priority 2
$\lambda^{0}_{i,1}+\lambda^{0}_{i,2}+\ldots+\lambda^{0}_{i,K}$
Buffer of priority P

Fig. 6. Structure of priority queue model.

$\lambda^p_{i,1}, \lambda^p_{i,2}, \ldots, \lambda^p_{i,K}$ denote the arriving rate of the streams whose packets are redecoded at the $(p-1)$th time. Recall that if a packet fails to decode, it is stored in the buffer waiting for the next parity symbol for recovery. As indicated in [14], if traffic stream A follows Poisson distribution, and each packet in A with some probability gets selected to generate a new traffic stream B, then packet stream B will also follow Poisson distribution. Therefore, the redecoded streams, which are "generated" by failed decoded packets, follow Poisson distribution and their arrival rates satisfy the following condition:

$$\lambda^p_{i,k}(n_{i,k}) = \lambda^1_{i,k} \prod_{\text{t}=1}^{p-1} \left(1 - F_\alpha(n_{i,k}, m_{\text{t}})\right) \tag{22}$$

where $n_{i,k}$ $(1 \leq k \leq U)$ denotes the number of hops in the route segment where kth traffic stream has traveled through since its last en/decoding in the route. Assuming that $\lambda^1_{i,k}$ and m_{t} have been predetermined, the value of $\lambda^p_{i,k}(n_{i,k})$ is determined by $n_{i,k}$. We assign priority p to the packet stream of $\lambda^p_{i,k}$ $(p = 1, 2, \ldots, P)$. The packets in a queue with the highest priority P enter the server (decoding and encoding part) first. If the queue of priority P is empty, then the packets of priority $P-1$ enter the server, and so on.

Assuming there are K data streams in the pth $(1 \leq p \leq P)$ priority queue, because each packet stream follows Poisson distribution, all of these streams can be combined into one stream $\lambda^p_i = \sum_{k=1}^{K} \lambda^p_{i,k}$. We use ρ_l to represent the *utilization* of a server when the first packet in the buffer with priority l enters the server and use Y_l to represent service time for a packet in a queue with priority l [20]. Recall $\mathbf{n}_i$ is the set of all $n_{i,k}$ $(1 \leq k \leq K)$. Then, $\rho_l(\mathbf{n}_i)$ can be calculated as

$$\rho_l(\mathbf{n}_i) = \overline{Y_l} \times \sum_{k=1}^{K} \lambda^l_{i,k}(n_{i,k}). \tag{23}$$

W_l represents the average delay of packets with priority l packets, and W_0 represents the average delay for one tagged waiting packet due to a packet already in service. W_0 can be calculated as

$$W_0(\mathbf{n}_i) = \sum_{l=1}^{P} \frac{\overline{Y_l^2}}{2\overline{Y_l}} \times \rho_l(\mathbf{n}_i). \tag{24}$$

As a result, the waiting time for each of the packets is

$$W_{\text{p}}(\mathbf{n}_i) = \frac{W_0(\mathbf{n}_i) + \sum\limits_{l=\text{p}+1}^{P} \rho_l(\mathbf{n}_i) W_l(\mathbf{n}_i)}{1 - \sum\limits_{l=\text{p}}^{P} \rho_l(\mathbf{n}_i)}. \tag{25}$$

From Kleinrock's conservation theorem in priority queuing model [25], the expected queuing delay for one packet in any node can be calculated as

$$\overline{W_{\text{que}}(\mathbf{n}_i)} = \sum_{\text{p}=1}^{P} \rho_{\text{p}}(\mathbf{n}_i) W_{\text{p}}(\mathbf{n}_i) = \frac{\rho(\mathbf{n}_i) W_0(\mathbf{n}_i)}{1 - \rho(\mathbf{n}_i)} \tag{26}$$

where

$$\rho\left(\mathbf{n}_i\right) = \sum_{\mathrm{p}=1}^{P} \rho_{\mathrm{p}}\left(\mathbf{n}_i\right). \tag{27}$$

Now, we consider the queueing delay for one packet, which might enter the queueing system several times due to re-en/decoding. During time interval T (T is large enough), the total number of packets N_{total} that enter the queueing system equals

$$N_{\text{total}}\left(\mathbf{n}_i\right) = \sum_{l=1}^{P}\sum_{j=1}^{K} \lambda_{i,j}^{l}\left(n_{i,j}\right) \times T. \tag{28}$$

The total waiting time can be given by

$$W_{\text{total}}\left(\mathbf{n}_i\right) = N_{\text{total}}\left(\mathbf{n}_i\right) \times \overline{W_{\text{que}}\left(\mathbf{n}_i\right)}. \tag{29}$$

Lemma 3.4: The expectation of the total queuing time for one packet when it goes through a node with $\mathbf{n}_i = [n_{i,1}, n_{i,2}, n_{i,3}, \ldots, n_{i,K}]$ can be calculated as

$$\begin{aligned}\overline{D_{\mathrm{q}}\left(\mathbf{n}_i\right)} &= \frac{W_{\text{total}}\left(\mathbf{n}_i\right)}{\sum\limits_{k=1}^{K} \lambda_{i,k}^{1}\left(n_{i,k}\right)} \\ &= \frac{\sum\limits_{\mathrm{p}=1}^{P}\sum\limits_{k=1}^{K} \lambda_{i,k}^{p}\left(n_{i,k}\right) \times \overline{W_{\text{que}}\left(\mathbf{n}_i\right)}}{\sum\limits_{k=1}^{K} \lambda_{i,k}^{1}}.\end{aligned} \tag{30}$$

If each receiver uses EEC to estimate the number of corrupted symbols, then the packet only needs to be retransmitted at most twice, i.e., $P = 2$. Hence, the expectation of the total queuing time equals

$$\overline{D_{\mathrm{q}}\left(\mathbf{n}_i\right)} = \frac{\sum\limits_{\mathrm{p}=1}^{2}\sum\limits_{k=1}^{K} \lambda_{i,k}^{p}\left(n_{i,k}\right) \times \overline{W_{\text{que}}\left(\mathbf{n}_i\right)}}{\sum\limits_{k=1}^{K} \lambda_{i,k}^{1}}. \tag{31}$$

The above priority queue model assumes that packets have no delay constraints, so it gives a higher priority to a packet that has been retransmitted more times. With the consideration of packet delay constraints (deadlines), we can first give a higher priority to the packet with smaller remaining time period to its deadline; if two packets have the same remaining time periods, we give a higher priority to the packet that has been transmitted more times. Modeling such a two-level priority queue to calculate queuing delay is nontrivial, so we leave this task as our future work.

E. *Minimizing the Delays*

As shown in Section II, we need to minimize $\overline{D_{\mathrm{q}}\left(\mathbf{n}_i\right)}$ and $\overline{D_{\text{p\&t}}\left(\mathbf{n}_i\right)}$ in order to achieve the objective of CEDAR in (4). According to (19), the prop&tran delay of the packet stream for $\mathbf{r}_k$ in v_i is calculated as

$$\overline{D_{\text{p\&t}}\left(n_{i,k}\right)} = \sum_{\mathrm{t}=1}^{\infty} f_G^{t}\left(n_{i,k}\right) D_{\text{p\&t}}^{t}\left(n_{i,k}\right). \tag{32}$$

Consequently, the average prop&tran delay of all the packet stream decoded in v_i is calculated as

$$\overline{D_{\text{p\&t}}\left(\mathbf{n}_i\right)} = \frac{\sum\limits_{k=1}^{K}\left[\lambda_{i,k}\sum\limits_{\mathrm{t}=1}^{\infty} f_G^{t} D_{\text{p\&t}}^{t}\left(n_{i,k}\right)\right]}{\sum\limits_{k=1}^{K} \lambda_{i,k}} \tag{33}$$

where $\mathbf{n}_i = [n_{i,1}\ n_{i,2}\ \ldots\ n_{i,K}]$. The average queuing delay of the packet stream in v_i can be derived from (30)

$$\overline{D_{\mathrm{q}}\left(\mathbf{n}_i\right)} = \frac{\sum\limits_{\mathrm{p}=1}^{P}\sum\limits_{k=1}^{K} \lambda_{i,k}^{p}\left(n_{i,k}\right) \times \overline{W_{\text{que}}\left(\mathbf{n}_i\right)}}{\sum\limits_{k=1}^{K} \lambda_{i,k}^{1}\left(n_{i,k}\right)}. \tag{34}$$

By minimizing the above $\overline{D_{\mathrm{q}}\left(\mathbf{n}_i\right)}$ and $\overline{D_{\text{p\&t}}\left(\mathbf{n}_i\right)}$, we retrieve two propositions presented below.

1) *Minimizing Queuing Delay:*

Proposition 3.1: Suppose $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and $\mathcal{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_K\}$ with total arrival rate $\sum_{k=1}^{K} \lambda_k = \lambda_{\text{total}}$, each packet is required to be decoded once in its route, and each node can decode the packet in any route, and also $\mathbf{n}_1 = \mathbf{n}_2 = \cdots = \mathbf{n}_N$. To minimize the total queueing delay for all the packets, the packet rate each node is responsible for should be the same. That is

$$\overline{\lambda_i} = \frac{\lambda_{\text{total}}}{N} \qquad (i = 1, 2, 3, \ldots, N). \tag{35}$$

Proof: According to (23), (24), (26), and (27), the queuing delay of packets decoded at v_i can be derived as

$$\overline{D_{\mathrm{q}}^{i}\left(\mathbf{n}_i\right)} = \frac{\rho\left(\mathbf{n}_i\right)}{1-\rho\left(\mathbf{n}_i\right)} \frac{\sum\limits_{l=1}^{P}\sum\limits_{k=1}^{B} \lambda_k^{l}\left(n_{i,k}\right)}{\sum\limits_{k=l}^{B} \lambda_k^{1}} W_0 = \frac{AJ\overline{\lambda_i}^2}{1-J\overline{\lambda_i}} \tag{36}$$

where A and J are invariants figured by (27), (23), and (22). Then, the following equation can be derived:

$$\sum_{i=1}^{N} \frac{AJ\overline{\lambda_i}^2}{1-J\overline{\lambda_i}} = -AC\lambda_{\text{total}} + \frac{AN}{J} + \frac{A}{J}\sum_{i=1}^{N}\frac{1}{\left(1-J\overline{\lambda_i}\right)}. \tag{37}$$

Let $H_i = 1-J\overline{\lambda_i}$, and then $\sum_{i=1}^{N} H_i = N - J\lambda_{\text{total}}$. According to Cauchy–Schwarz inequality [14], we find that

$$\begin{aligned}\sum_{i=1}^{N}\frac{N-J\lambda_{\text{total}}}{H_i} &= \sum_{i=1}^{N}\frac{1}{H_i}\sum_{i=1}^{N} H_i \qquad (38)\\ &= \sum_{i=1}^{N}\frac{1}{\left(\sqrt{H_i}\right)^2}\sum_{i=1}^{N}\left(\sqrt{H_i}\right)^2 \\ &\geq \left(\sum_{i=1}^{N}\sqrt{\frac{1}{H_i}}\sqrt{H_i}\right)^2 = N^2 \qquad (39)\end{aligned}$$

from which, we can derive that

$$\sum_{i=1}^{N}\frac{1}{\left(1-J\overline{\lambda_i}\right)} = \sum_{i=1}^{N}\frac{1}{H_i} \geq \frac{N^2}{N-J\lambda_{\text{total}}} \tag{40}$$

and

$$\sum_{i=1}^{N}\frac{AJ\overline{\lambda_i}^2}{1-J\overline{\lambda_i}} \geq -A\lambda_{\text{total}} + \frac{AN}{J} + \frac{AN^2}{J\left(N-\lambda_{\text{total}}\right)} \tag{41}$$

which reaches its minimum value when $1-J\overline{\lambda_1} = 1-J\overline{\lambda_2} = \cdots = 1-J\overline{\lambda_N}$, or the values of $\overline{\lambda_i}$ $(i = 1, 2, .., N)$ are equal to each other. □

According to Proposition 3.1, given several packet streams and number of nodes required to decode these packets, we need to balance the en/decoding load for all of these nodes. For example, in Fig. 7, three packet streams λ_1, λ_2, and λ_3 are transmitted through the 1st route, the 2nd route and the 3rd route,

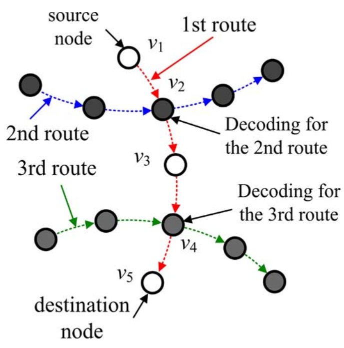


Fig. 7. Example of balancing en/decoding load.

respectively. The 2nd route chose v_2 as the key node, and the 3rd route chose v_4 as the key node. For the 1st route, v_1 is the packet source (*sender*), and v_5 is the packet destination (*receiver*). It needs to choose one node among intermediate nodes v_2, v_3, and v_4 to decode packets. To achieve load balance and hence reduce queuing delay, its best choice should be v_3 since v_2 and v_4 are already responsible for en/decoding for the 2nd and 3rd routes, respectively.

2) Minimizing Prop&tran Delay: Consider a route with n hops that is not affected by the interference from any other routes. Our objective is to divide it into several route segments with the size $n_1, n_2, \ldots, n_e$, respectively, in order to minimize the total delay of packet stream transmission. We consider one of these route segments that has n_k hops, and use $\overline{D^{\text{ave}}_{\text{p\&t}}}$ to denote the average prop&tran delay for each hop in this route segment. That is

$$\overline{D^{\text{ave}}_{\text{p\&t}}(n_i)} = \frac{\overline{D_{\text{p\&t}}(n_i)}}{n_i} \qquad (i = 1, 2, 3, \ldots, e) \tag{42}$$

where $\overline{D^{\text{ave}}_{\text{p\&t}}(n_i)}$ is a function of n_i. We use $\overline{D^{\text{ave}}_{\text{p\&t,min}}}$ to represent the minimized value of $\overline{D^{\text{ave}}_{\text{p\&t}}(n_i)}$, and we need to search n_{opt} that satisfies $\overline{D^{\text{ave}}_{\text{p\&t}}(n_{\text{opt}})} = \overline{D^{\text{ave}}_{\text{p\&t,min}}}$,

Proposition 3.2: To divide one route into several route segments, the optimal length for each route segment should be n_{opt} in order to minimize the prop&tran delay for the packet delivery.

Proof: The sum of prop&tran delay for the e route segments equals

$$\sum_{i=1}^{e} \overline{D_{\text{p\&t}}(n_i)} = \sum_{i=1}^{e} \frac{\overline{D_{\text{p\&t}}(n_i)}}{n_i} \times n_i$$
$$\geq \overline{D^{\text{ave}}_{\text{p\&t,min}}} \times \sum_{i=1}^{e} n_i = \overline{D^{\text{ave}}_{\text{p\&t,min}}} \times n. \tag{43}$$

When $n_1 = \cdots = n_e = n_{\text{opt}}$, $\sum_{i=1}^{e} \overline{D_{\text{p\&t}}(n_i)}$ reaches its minimum value. □

F. Balancing of En/Decoding Load

In some wireless network applications (e.g., wireless sensor networks), balanced en/decoding load distribution among all nodes greatly affects the network performance including the connectivity and lifetime of the network. Thus, besides minimizing the total packet delay of the system, we have the secondary objective, which is to minimize the difference of the en/decoding load of all the nodes. Recall that "en/decoding load" of a node, say v_i, is defined as the number of packets v_i needs to en/decode in a time unit. According to the queuing model built in Section III-D, v_i needs to en/decode all the packets waiting the priority queue from different routes, and we can derive the value of $\overline{L(\mathbf{n}_i)}$ from (22)

$$\overline{L(\mathbf{n}_i)} = \sum_{k=1}^{K} \sum_{p=1}^{P} \lambda_{i,k}^{p}(n_{i,k})$$
$$= \sum_{k=1}^{K} \sum_{p=1}^{P} \lambda_{i,k}^{1} \prod_{t=1}^{p-1} (1 - F_\alpha(n_{i,k}, m_t)). \tag{44}$$

We use $\overline{L(\mathbf{N})} = \frac{\sum_{i=1}^{N} \overline{L(\mathbf{n}_i)}}{N}$ to represent the average en/decoding load of all the nodes in $\mathcal{V}$. Since standard deviation [14] of en/decoding load (represented by $\sigma(\mathbf{N})$) shows how much variation or "dispersion" exists from the average $\overline{L(\mathbf{N})}$, we use $\sigma(\mathbf{N})$ to measure the fairness of en/decoding load of all the nodes: The lower the value of $\sigma(\mathbf{N})$, the higher the fairness of the en/decoding load of all the nodes. $\sigma(\mathbf{N})$ is defined by

$$\sigma(\mathbf{N}) = \sqrt{\sum_{i=1}^{N} \left(\overline{L(\mathbf{n}_i)} - \overline{L(\mathbf{N})}\right)^2}. \tag{45}$$

Then, the objective is to minimize $\sigma(\mathbf{N})$.

Proposition 3.3: Suppose there are N nodes and B routes with total arrival rate λ_{total} and each packet is required to be decoded once in its route, and each node can decode the packet in any route with $\mathbf{n}_1 = \mathbf{n}_2 = \mathbf{n}_3 = \cdots = \mathbf{n}_N$. To minimize $\sigma(\mathbf{N})$, the en/decoding load of each node should be the same.

Proof: When the packet arrival rate that each node v_i is responsible for is $\overline{\lambda_i} = \frac{\lambda_{\text{total}}}{N}$, according to (44), we can derive that $\overline{L(\mathbf{n}_1)} = \cdots = \overline{L(\mathbf{n}_N)} = \overline{L(\mathbf{N})}$, which indicates that $\sigma(\mathbf{N}) = 0$. Because $\sigma(\mathbf{N}) \geq 0$, we can conclude that when the en/decoding load of each node is same, the value of $\sigma(\mathbf{N})$ is minimized. □

Propositions 3.1 and 3.3 imply that minimizing the queuing delay and balancing en/decoding load distribution among decoding nodes in the network can be achieved simultaneously.

IV. Scalable and Distributed Scheme

The objective function of CEDAR in (4) is a nonlinear integer programming (NIP) problem. Solving this problem leads to minimizing the total delay for all the packets in the network. This, however, requires each node to collect a global knowledge of the network including the routes and the arrival rate for each traffic stream, which is nearly impractical in wireless applications such as wireless ad hoc networks. Even though the global knowledge is available, the problem is NP-hard as it is an NIP problem [26]. Thus, we need to design a scalable and distributed scheme for identifying the key nodes for each route.

Simply put, Propositions 3.1 and 3.3 indicate that the scheme should try to balance the en/decoding load of each node to minimize the queuing delay; Proposition 3.2 indicates the optimal route segment length (i.e., the positions of key nodes) to minimize the prop&tran delay. If both requirements can be satisfied simultaneously, the scheme will satisfy the objective function. However, these two requirements may conflict with each other. We identify different network traffic load situations that each proposition should be primarily considered, and also propose a method to coordinately consider these two propositions when choosing key nodes.

Algorithm 1: Identify key nodes in route **r** executed by each node in **r** in a light-traffic network.

```
begin
    Set SEN_FIN, REC_FIN and DEC to 0 ;
    while $SEN\_FIN = 0$ or $REC\_FIN = 0$ do
        Listen to other nodes;
        if it has received ACK_REC from the next node in r then
            $\mathrm{SEN\_FIN} \leftarrow 1$;
        if it receives (OPT_HOP, FLAG) from the previous node in r then
            $\mathrm{REC\_FIN} \leftarrow 1$;
            Send ACK_REC to the previous node;
            if $FLAG = 0$ then
                $\mathrm{DEC} \leftarrow 1$ // It is a key node;
                $\mathrm{FLAG} \leftarrow \mathrm{OPT\_HOP}$;
            else
                $\mathrm{FLAG} \leftarrow \mathrm{FLAG} - 1$;
```

A. Case I (Light Traffic)

When a wireless network has light traffic, because the influence from queuing delay is much less significant, and on average en/decoding load of each node is low, we mainly consider the prop&tran delay. As Proposition 3.2 indicates, we first search the value of n_{opt} and then set $\mathrm{OPT\_HOP} = n_{\mathrm{opt}}$, and set $\mathrm{FLAG} = \mathrm{OPT\_HOP}$. In a routing algorithm [27], every node keeps a routing table, and a source node sends out a message to find the route to a destination for transmitting a packet stream. After a source–destination route has been discovered, each node in the route determines whether it is a key node in a distributed manner by executing the key node identification algorithm. Algorithm 1 presents the pseudocode of this algorithm executed by every node (except source node and destination node), say v, in a route **r** in the case of light network traffic. Basically, the source node sends a packet (OPT_HOP, FLAG) through the route and nodes. FLAG is decreased by one in each hop, and the node receiving the packet with $\mathrm{FLAG} = 0$ is a key node. This node then restores $\mathrm{FLAG} = \mathrm{OPT\_HOP}$ before forwarding the packet to the next node. Here, SEN_FIN presents whether v has received ACK from the next node in **r**; REC_FIN presents whether v has received (OPT_HOP, FLAG) from the previous node in **r**; DEC presents whether v is responsible for en/decoding;

B. Case II (Heavy Traffic)

When a wireless network has heavy traffic, we aim to reduce queuing delay while reducing the prop&tran delay. Also, we need to decrease the difference of en/decoding load overall the network. Fortunately, according to Propositions 3.1 and 3.3, decreasing queuing delay and increasing load balancing of the network can be achieved simultaneously.

When a new route is built, the CEDAR scheme first executes Algorithm 1. When executing the algorithm, each node along the route piggybacks its en/decoding load to the packet, and the last node sends the collected en/decoding load information to the source node. The source node then knows the series of nodes identified as "potential key nodes" and their en/decoding load and calculates the average en/decoding load through the route, denoted as AVE_LOAD. It then checks whether the en/decoding load of each identified key node is larger than a predefined threshold (AVE_LOAD + BOUND), where BOUND is a predetermined value. An overloaded potential key node is replaced by the node closest to itself in the route that has load below the threshold. We use LOAD_i to denote the en/decoding load of ith node in the route. If $\mathrm{LOAD}_i > (\mathrm{AVE\_LOAD} + \mathrm{BOUND})$, then the source node compares LOAD_{i-1} and LOAD_{i+1}, and chooses the node with smaller en/decoding load if the en/decoding load is smaller than the threshold. If both LOAD_{i-1} and LOAD_{i+1} are larger than the threshold, the source node compares LOAD_{i-2} and LOAD_{i+2}. The source node repeats this process until finding a node with load below the threshold or the next two nodes needs to be compared are out of the range of $[i - \lfloor \mathrm{OPT\_HOP}/2 \rfloor, i + \lfloor \mathrm{OPT\_HOP}/2 \rfloor]^{\mathrm{th}}$. The pseudocode of this algorithm is presented in Algorithm 2. In order to release the key node selection load on the source node, this algorithm can be easily extended to a fully decentralized algorithm. Specifically, the destination node calculates the AVE_LOAD and forwards it back to the source node along the route. Each node checks whether its own load is beyond the threshold. If so, it probes its nearby nodes sequentially until finding a node with load within the threshold or meeting an identified potential key node.

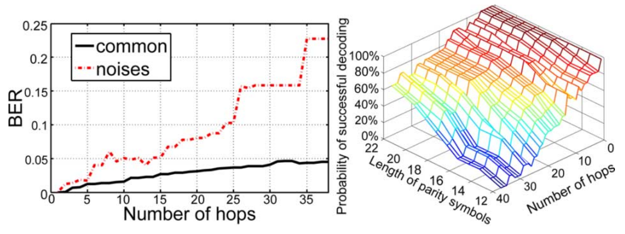
In dynamic wireless networks, the network topology and the packet arrival rates change over time, which require nodes to recalculate the key nodes periodically. Based on Algorithms 1 and 2, we see that such dynamics only affect the route length and the en/decoding load of each node in the route that are needed in key node calculation. Thus, to deal with the dynamics, we let the sender periodically send a packet through the route to probe this information. As a result, the key nodes calculated by a packet sender are the correct key nodes for the current network environment, and CEDAR is adaptive to the network dynamics.

V. Performance Evaluation

This section presents the results of experiments on NESTbed [28] and simulation with MATLAB. We compared CEDAR to the global optimal solution (OPTIMAL), the traditional link-layer protocols, where a packet is decoded hop by hop [6], [16] (HBH), and to another solution where a packet is only decoded at the destination (DEST). In order to evaluate the effect of the load balancing algorithm (Algorithm 2) in CEDAR, we also test the performance of CEDAR without this algorithm denoted by CEDAR*. We measured the following metrics: 1) *packet delay*: the time interval from the time a packet is generated by the source node to the time that the packet is successfully decoded at the destination node; 2) *throughput*: the total number of data bits successfully decoded at the destinations per time unit (millisecond) in the entire network; 3) *en/decoding load*: the number of packets a node en/decodes; 4) *number of probing messages*: the total number of messages used for probing.

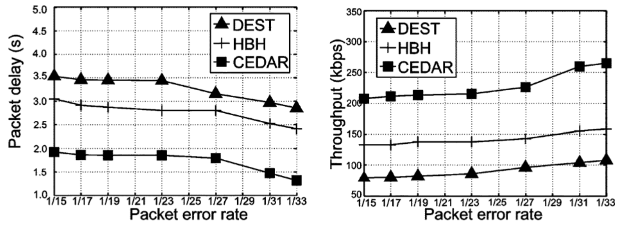
A. Experiments on Real-World NESTbed

NESTbed is an open testbed for developing wireless sensor systems [28]. It is a collection of 80 TELOSB sensors that are arranged in a grid. The sensors have a CC2420 Chip and communicate using the IEEE 802.15.4 standard. We verify our mathematical models and evaluate the performance of CEDAR on NESTbed. We created a multihop network of TELOSB sensors



(a) (b)

Fig. 8. Comparison of real-world NESTbed results. (a) BER in multiple hops. (b) Successful decoding rate.



(a) (b)

Fig. 9. Experimental results on real-world NESTbed. (a) Packet delay. (b) Throughput.

Algorithm 2: Select key nodes with consideration of load balance in a heavy-traffic or normal-traffic network.

```
begin
    Use Algorithm 1 to get the "potential key nodes" in route **r**;
    Let node $i$ be one node selected as the "potential key node";
    $j = 0$;
    **while** $j \leq \lfloor \mathrm{OPT\_HOP}/2 \rfloor$ **do**
        **if** $\mathrm{LOAD}_{i+j} \leq \mathrm{LOAD}_{i-j}$ **then**
            **if** $\mathrm{LOAD}_{i+j} \leq (\mathrm{BOUND} + \mathrm{AVE\_LOAD})$ **then**
                **return** $i + j$;
        **else**
            **if** $\mathrm{LOAD}_{i-j} \leq (\mathrm{BOUND} + \mathrm{AVE\_LOAD})$ **then**
                **return** $i - j$;
        $j = j + 1$;
    **return** $0$;
```

running Tiny-OS 2.1.0 written in NESC. We use Reed–Solomon codes to detect and fix errors in a packet; if the number of error bits exceeds the capability of Reed–Solomon codes for correction, the receiver asks for retransmission.

Mathematical Model Verification: We measured BER after the packet traveled for different numbers of hops in two scenarios, *common* and *noises*. Fig. 8(a) shows BER versus the number of hops. We see that BER increases as the number of hops increases in both scenarios. This is because as the number of hops increases, more flipped bits are generated, and errors tend to be cumulative and propagated along a routing path in the multihop network. Also, *common* produces much lower BER than *noises* as more noises increase BER.

We then measured the probability of successful decoding when the error correction node [responsible of error correction code (ECC)] is away from the source node by a different number of hops and when the parity symbols have different lengths. Fig. 8(b) shows the probability of successful decoding in a multihop network with varying parity symbol lengths and the number of hops between the error correction node and the source node. The figure illustrates that if the error correction node is farther away from the source, the probability of a packet drop increases. Also, when the length of parity symbols increases, the probability of successful decoding increases, which is consistent with (13). Furthermore, when the number of hops is large, increasing the length of parity symbols will not guarantee the successful decoding of a packet. The experimental results indicate that in order to maintain a high throughput and low packet drop rate, it is necessary to fix errors in the packet as soon as possible. Thus, arranging the destination to decode the packets is not an effective method. However, performing retransmission or ECC at every hop is not efficient due to queuing delays. An effective method to decode a packet is to fix errors in the intermediate nodes, which can successfully decode the packet while increasing the efficiency.

Scheme Performance Evaluation: In the experiments on NESTbed, we chose eight sources and eight destinations, and the source–destination path length was 75 hops. Given that data rate $R = 250$ kb/s, average $\mathrm{BER} \approx 0.001$ [common scenario in Fig. 8(a)], $L_{\mathrm{pac}} = 50$ B, and $L_{\mathrm{ACK}} = 28$ B, we calculate the optimal number of hops one packet should be decoded once is 9 and 10.

Each source node generated 500 packets at a time interval varying from 80 to 160 ms. We measured the delay of transmissions and the throughput, which is defined as the total size of all the packets divided by the total time used for transmitting all the packets. The packet error rate is defined as the average percent of unsuccessfully transmitted packets in each hop. To test the performance of the three schemes in different environments, we manually changed the packet error rate from 1/15 to 1/33. Fig. 9(a) shows the packet delay of CEDAR, DEST, and HBH. We find that the average packet delay of CEDAR is much lower than that of DEST and HBH. DEST has the highest delay because it assigns the decoding work to each packet's destination rather than the intermediate nodes, which generates much higher probability of packet redecoding due to higher probability of packet errors, thus increasing the delay. The delay of HBH is higher than that of CEDAR because HBH requires packets to be en/decoded in each hop, which generates high en/decoding load on intermediate nodes, leading to high queuing delay. Fig. 9(b) shows the throughput of three schemes. From the figure, we can find that the throughput follows $\mathrm{CEDAR} > \mathrm{HBH} > \mathrm{DEST}$. This is because lower packet transmission delay usually leads to higher throughput in the network.

B. Simulation on MATLAB

We conducted simulation on MATLAB to evaluate the performance of CEDAR. We built a 9×9 grid network with each node located in one grid and randomly selected 16 pairs of source node and destination node. Each packet contains 20 data symbols, five type-I parity symbols, and five type-II parity symbols. Each symbol has 5 bits. For the scheme using EEC code, there are $\lfloor \log_2 30 \rfloor = 4$ levels of EEC bits in each packet, with 5 EEC bits in each level. Also, we randomly chose nodes connecting each pair of source node and destination node as the route. We

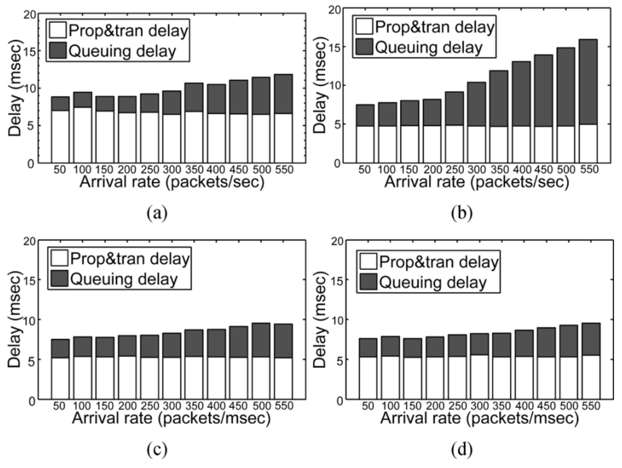


Fig. 10. Comparing prop&tran delay and queuing delay. (a) DEST. (b) HBH. (c) CEDAR*. (d) CEDAR.

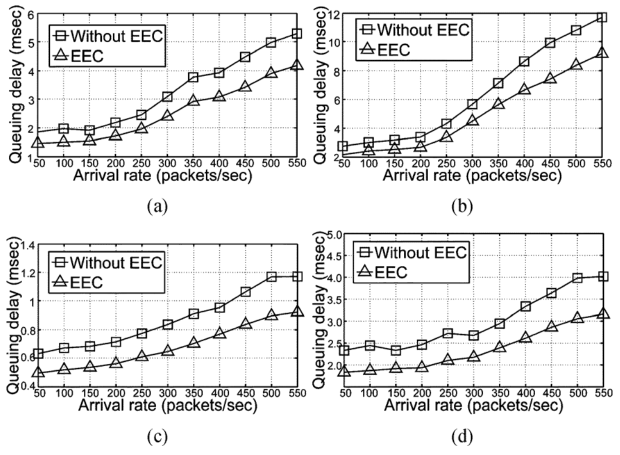


Fig. 11. Queuing delay with and without using EEC. (a) DEST. (b) HBH. (c) CEDAR*. (d) CEDAR.

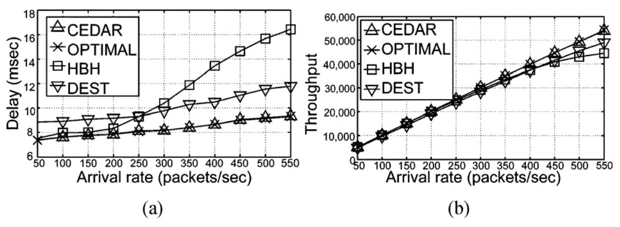


Fig. 12. Packet delay and throughput (with OPTIMAL). (a) Average packet delay. (b) Throughput.

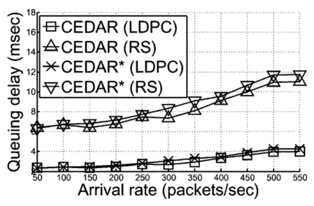


Fig. 13. RS versus LDPC.

use low density parity check (LDPC) code [29] for en/decoding packets.

Fig. 10(a)–(d) compares prop&tran delay and queuing delay computed by HBH, DEST, CEDAR*, and CEDAR, respectively. From the figures, we can find that: 1) the queuing delay increases as the generating rate of each data stream increases but the prop&tran delay remains nearly constant; 2) the queuing delay increases more significantly in HBH than in DEST and CEDAR (i.e., it follows $\text{CEDAR} < \text{DEST} < \text{HBH}$); 3) for prop&tran delay, it follows $\text{HBH} < \text{CEDAR} < \text{DEST}$; 4) CEDAR generates the same prop&tran delay but lower queuing delay than CEDAR*; and 5) the total packet delay follows $\text{CEDAR} < \text{CEDAR}^* < \text{DEST} < \text{HBH}$. For 1), this is because queuing delay is determined by the generating rate of the source node, but the prop&tran delay is independent of it. For 2), 4), and 5), HBH has higher queuing delay since it generates more en/decoding load on intermediate nodes. In contrast to HBH, DEST only assigns the decoding work to each packet's destination, which increases both prop&tran delay and queuing delay due to higher probability of packet redecoding [as (13) shows]. Instead of accumulating decoding work on the destinations, CEDAR* and CEDAR choose a number of intermediate nodes to be responsible for the en/decoding work to reduce the probability of redecoding. CEDAR performs better than CEDAR* because CEDAR distributes the en/decoding load of the intermediate nodes more evenly, which reduces the queuing delay as indicated in Proposition 3.1.

Fig. 11(a)–(d) compares the queuing delays of the four schemes with and without using EEC versus different packet arrival rates. From the figures, we see that using EEC, the queuing delays of the four schemes are reduced significantly. This result is caused by two reasons. First, EEC decreases the number of retransmissions by estimating BER of the packets. Second, the computing time of EEC is much less than erasure code (e.g., Reed–Solomon code [19]) according to (30) and (31). These experimental results confirm the effectiveness of EEC in enhancing the performance of CEDAR. Fig. 12(a) and (b) compares OPTIMAL to CEDAR, CEDAR*, DEST, and HBH in terms of *packet delay* and *throughput*. Considering NP-hard feature of the problem, we only set a small-scale network (six source nodes and six destination nodes). The results demonstrate that CEDAR can achieve almost the "best" performance in terms of packet delay even in the distributed manner. Fig. 13 compares the queuing delay of CEDAR and CEDAR* using LDPC code and Reed–Solomon code [29], which is also a widely used code for HARQ protocol. From the figures, we find that the schemes using LDPC has much smaller queuing delay than those using Reed–Solomon code. Though LDPC has a chance of failure, LDPC is much faster than Reed–Solomon code for en/decoding packet (in our simulation on MATLAB, to en/decode a packet, LDPC code is about 1.8 times faster than Reed–Solomon code). Accordingly, the en/decoding time (service time) for LDPC is smaller, which leads to the queuing delay of LDPC being smaller [according to (30)].

Fig. 14(a) and (b) compares the packet delay of CEDAR* and CEDAR in the static and dynamic scenarios, respectively. In the dynamic scenario, due to the network topology change, the nodes in a route of each sender were changed to other randomly selected nodes once per second, i.e., dynamic rate was set to 1 times/s. The sender probing frequency was equal to the dynamic rate. From both figures, we find that CEDAR* and

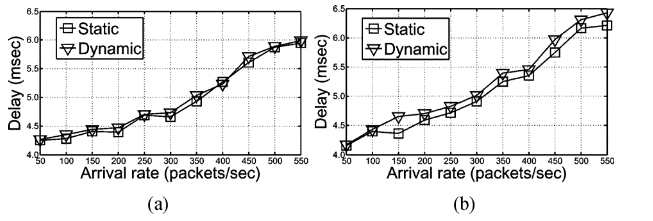


Fig. 14. Effect of dynamics with different arrival rates. (a) CEDAR. (b) CEDAR*.

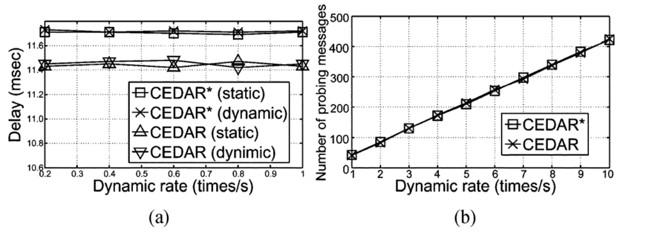


Fig. 15. Effect of dynamics with different dynamic rates. (a) Delay. (b) Number of probing messages.

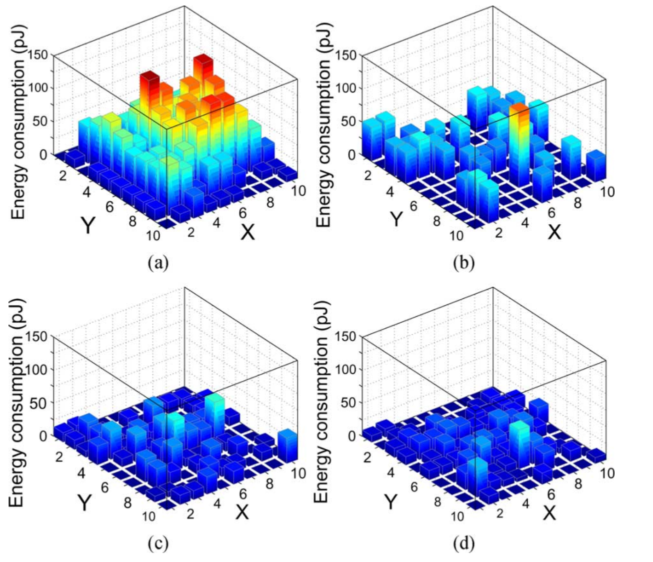


Fig. 16. En/decoding load distribution of all the nodes. (a) HBH. (b) DEST. (c) CEDAR*. (d) CEDAR.

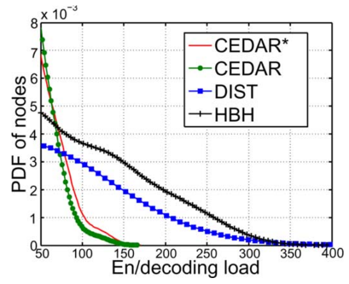


Fig. 17. Distribution.

CEDAR achieve almost the same performance in static scenario and dynamic scenario. This is because in both CEDAR* and CEDAR, the senders periodically send the packet to probe the information through the routes, so they always know the most updated information to determine the en/decoding routing nodes even in the dynamic network. Fig. 15(a) and (b) shows the packet delay and the number of probing messages versus the dynamic rate, respectively. From Fig. 15(a), we find that the packet delays of both CEDAR and CEDAR* remain at the same level as the dynamic rate increases. Fig. 15(b) shows that the number of probing messages increases as the dynamic rate increases since each sender must update information more frequently when the dynamic rate becomes higher.

C. Load Balancing

Fig. 16(a)–(d) shows the en/decoding load of each node in HBH, DEST, CEDAR*, and CEDAR, respectively. The coordinate (x, y) $(1 \le x \le 9, 1 \le y \le 9)$ in the figures shows the position of each node in the network grid. We observe that most nodes in HBH have high decoding load, and the decoding loads among nodes vary greatly. DEST generates lower decoding load on nodes and move balance load distribution than HBH. Most nodes in CEDAR have much lower decoding load than nodes in HBH and DEST. From our experimental result data, we find that the average en/decoding load of HBH, DEST, and CEDAR per node is approximately 90, 32, 24, and 24. $(\mathrm{HBH} > \mathrm{DEST} > \mathrm{CEDAR}^* \approx \mathrm{CEDAR})$, respectively, and their standard deviations are 78, 49, 28, and 24 $(\mathrm{HBH} > \mathrm{DEST} > \mathrm{CEDAR}^* > \mathrm{CEDAR})$, respectively. Thus, CEDAR can better balance the en/decoding load among nodes compared to the other two schemes and CEDAR*, and also decreases the average decoding load on a node compared to DEST and HBH. The better performance of CEDAR* than HBH and DEST confirms the effectiveness of CEDAR's intermediate node selection algorithm in distributing load among partial intermediate nodes rather than all intermediate nodes or only the destination nodes. The better performance of CEDAR over CEDAR* confirms the effectiveness of CEDAR's load balancing algorithm in balancing the en/decoding load among nodes and hence reducing the queuing delay. Unlike HBH that arranges every node through a route to conduce en/decoding regardless of its current en/decoding load, DEST only assigns the destination for decoding, thus reducing decoding load on nodes. CEDAR only arranges the intermediate nodes in a route that has available en/decoding capacity to be decoders, which constrains the en/decoding load of intermediate nodes. CEDAR performs better than DEST in terms of the assigned decoding load per node and load balance. Recall that (14) indicates the probability of successful decoding decreases very rapidly if the number of hops between a decoding node and its previous decoding node increases. More decoding failures for a packet in DEST lead to more decoding operations, hence higher decoding load. Unlike DEST that biases on the destinations for the decoding operations, CEDAR distributes the decoding load among intermediate nodes that have available capacity for decoding, achieving more balanced load distribution.

Fig. 17 shows the probability density function (PDF) of nodes based on en/decoding load in CEDAR, DEST, CEDAR*, and HBH. We see that the en/decoding load of nodes in HBH is higher and more unevenly distributed than DEST and CEDAR. The reason for HBH's inferior performance is that it makes all the intermediate nodes be responsible for en/decoding all the packets going through them, which generates high en/decoding load on these nodes and also greatly increases the en/decoding load in the network. Though the mean value of the en/decoding load of DEST is not very high, the load among nodes is not distributed in balance and some nodes have en/decoding load larger than 250. By always conducting decoding in the destination nodes that are randomly selected from the network, DEST distributes the load more evenly than HBH. However, the longer distance between the decoding node and encoding node increases the probability of packet errors and recovery failures, leading to more redecoding operations. Therefore, DEST generates more en/decoding load than CEDAR.

VI. Related Work

Error detection and correction is one the richest problems in communication literature. The link-layer protocol of the current TCP/IP stack has adopted variations of error recovery mechanisms to provide reliability for point-to-point communication especially for wireless systems. Different wireless communication standards currently utilize variations of error control protocols that generally can be categorized into ARQ [13] and HARQ-based [5], [11] protocols. For instance, IEEE802.11 WiFi uses ARQ where a receiving node discards corrupted packets (even when there is only a single bit error) and requests for a retransmission. The 4G/LTE deploys HARQ with Turbo Codes where the sender node encodes the packet payload using Turbo channel codes [29] prior to the transmission. Accordingly, the receiver node requests for a retransmission when the decoding of the received packet fails.

In conjunction with the current wireless link-layer standards, there is significant work and research conducted to improve the performance of either ARQ- or HARQ-based protocols. Several kinds of HARQ protocols (see [5], [11], and the references therein) improve the throughput of the ARQ schemes by packet combining, e.g., by keeping the erroneous received packets and utilizing them for detection and packet recovery. Examples of recent efforts for combating the inefficiency of ARQ-based wireless protocols include partial packet recovery (PPR) [9], SOFT [3], and automatic code embedding (ACE) framework [16]. Some of these approaches, such as PPR and SOFT, exploit physical-layer information regarding the quality of individual bits to increase the probability of recovering corrupted packets. Other schemes, such as ACE, utilize information available in the current 802.11 link-layer protocols in conjunction with error correcting codes to recover corrupted packets. Ilyas *et al.* [30] proposed the “Poor Man’s SIMO System” (PMSS) to reduce packet losses in networks of commodity IEEE 802.15.4 sensor motes using cooperative communication and diversity combination. Based on mathematical analysis, Jelenkovi *et al.* [4] proposed a new dynamic packet fragmentation algorithm that can adaptively match channel failure characteristics. Reuven *et al.* [1] considered a new scenario, in which when a base station wishes to multicast information to a large group of nodes using application-layer FEC codes.

These aforementioned works have significantly improved the ARQ- and HARQ-based link-layer performance and provide a comprehensive error control approach for wireless communication. However, virtually all of these efforts follow the conventional TCP/IP link-layer “store-and-forward” design paradigm where each relay node verifies the correctness of each packet before forwarding it to the next node. This inherently introduces substantial overhead on bandwidth utilization and throughput and the overall end-to-end delay. In addition, the point-to-point error recovery is not an optimal approach for energy-constrained dense wireless networks. Though the previous work MIXIT [12] has jettisoned reliable link-layer error detection and recovery altogether using a symbol-level network coding, its coding/decoding algorithms are more demanding for computational capacities of nodes than traditional store-and-forward methods. Compared to MIXIT’s implementation on software radios, CEDAR is more suitable for the devices with constrained processing capability, e.g., sensors, because CEDAR implements the decoding process by Reed–Solomon, which can be encoded and decoded by hardware. Accordingly, in this paper, we pursue a paradigm shift in the conventional link-layer design and propose a distributed, low-complexity, and adaptive scheme to achieve high reliability, stability, and energy efficiency in packet transmission. CEDAR is introducing a new chapter in link-layer design for future wireless networks comprising energy-constrained nodes where error recovery is optimally conducted in selected nodes.

VII. Conclusion

In this paper, our objective is to find an optimal solution to choose immediate nodes in transmission routes for en/decoding packets in wireless networks in order to minimize the packet delay and increase the throughput. We mathematically analyze the packet delay and model the problem as an integer programming problem, which helps to discover a globally optimal solution. Taking into account the scalability of the network and limitation of the information that each node can collect, we propose a distributed scheme that can achieve performance comparable to the globally optimal solution. The simulation results in MATLAB demonstrate that our scheme performs better than previous packet recovery schemes. Since CEDAR does not have to be tied up with any specific error estimator, e.g., EEC, in our future work, we will try different error estimation codes for CEDAR. Also, we will further consider how to design the method to settle the problem that data rate may vary in each hop for transmissions. Finally, we will combine opportunistic routing with CEDAR to further reduce decoding delay, hence packet delay. That is, when a key node overhears a packet, it directly forwards it rather than waiting for the packet sent by its previous receiver to decode it.

References

[1] R. Cohen, L. Grebla, and G. Katzir, “Cross-layer hybrid FEC/ARQ reliable multicast with adaptive modulation and coding in broadband wireless networks,” in *Proc. IEEE INFOCOM*, 2009, pp. 1917–1925.

[2] Z. Guo *et al.*, “A practical joint network- channel coding scheme for reliable communication in wireless networks,” in *Proc. ACM MobiHoc*, 2009, pp. 279–288.

[3] G. Woo, P. Kheradpour, D. Shen, and D. Katabi, “Beyond the bits: Cooperative packet recovery using physical layer information,” in *Proc. MobiCom*, 2007, pp. 147–158.

[4] P. R. Jelenkovi and J. Tan, “Dynamic packet fragmentation for wireless channels with failures,” in *Proc. ACM MobiHoc*, 2008, pp. 73–82.

[5] E. C. Strinati, S. Simoens, and J. Boutros, “Performance evaluation of some hybrid ARQ schemes in IEEE 802.11a networks,” in *Proc. VTS*, 2003, pp. 2735–2739.

[6] K. C. Lin, N. Kushman, and D. Katabi, “ZipTx: Harnessing partial packets in 802.11 networks,” in *Proc. ACM MobiCom*, 2008, pp. 351–362.

[7] S. S. Karande and H. Radha, “Hybrid erasure-error protocols for wireless video,” *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 307–319, Feb. 2007.

[8] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability gain of network coding in lossy wireless networks," in *Proc. IEEE INFOCOM*, 2008, pp. 196–200.
[9] K. Jamieson and H. Balakrishnan, "PPR: Partial packet recovery for wireless networks," in *Proc. SIGCOMM*, 2007, pp. 409–420.
[10] S. Soltani, K. Misra, and H. Radha, "Delay constraint error control protocol for real-time video communication," *IEEE Trans. Multimedia*, vol. 11, no. 4, pp. 742–751, Jun. 2009.
[11] H. Yomo, S. S. Chakraborty, and R. Prasad, "PHY and MAC performance evaluation of IEEE 802.11a WLAN over fading channels," *IETE J. Res.*, vol. 51, no. 1, pp. 83–94, Jan.–Feb. 2005.
[12] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-level network coding for wireless mesh networks," in *Proc. ACM SIGCOMM*, 2008, pp. 401–412.
[13] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.
[14] S. Chahramant, *Fundamentals of Probability with Stochastic Process*. Reading, MA, USA: Eastern New England College Press, 1986.
[15] H. S. Wang and N. Moayeri, "Finite-state Markov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, Feb. 1995.
[16] S. Soltani, K. Misra, and H. Radha, "On link-layer reliability and stability for wireless communication," in *Proc. ACM MobiCom*, 2008, pp. 327–338.
[17] S. Wicker and V. Bhargava, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.
[18] M. Mandelbaum, M. Hlynka, and P. H. Brill, "Nonhomogeneous geometric distributions with relations to birth and death processes," *TOP*, vol. 15, pp. 281–296, 2007.
[19] B. Chen, Z. Zhou, Y. Zhao, and H. Yu, "Efficient error estimating coding: Feasibility and applications," in *Proc. ACM SIGCOMM*, 2010, pp. 3–14.
[20] J. L. Hammond and P. J. O'Reilly, *Performance Analysis of Local Computer Networks*. Reading, MA, USA: Addison-Wesley, 1988.
[21] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1597–1609, Dec. 2011.
[22] M. J. Neely, "Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1146–1159, Aug. 2009.
[23] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1481–1493, Oct. 2009.
[24] M. Becchi, "From Poisson processes to self-similarity: A survey of network traffic models," 2008.
[25] V. B. Iversen, *Teletraffic Engineering Handbook*. Kongens Lyngby, Denmark: Tech. Univ. Denmark Press, 2001.
[26] P. Tian, J. Ma, and D.-M. Zhang, "Non-linear integer programming by Darwin and Boltzmann mixed strategy," *Eur. J. Oper. Res.*, vol. 105, no. 1, pp. 224–235, Feb. 1996.
[27] L. L. Peterson and B. S. Davie, *Computer Network: A System Approach*. San Mateo, CA, USA: Morgan Kaufmann, 2007.
[28] A. R. Dalton and J. O. Hallstrom, "An interactive, source-centric, open testbed for developing and profiling wireless sensor systems," *Int. J. Distrib. Sensor Netw.*, vol. 5, no. 2, pp. 105–138, Mar. 2009.
[29] D. N. Rowitch and L. B. Milstein, "On the performance of hybrid FEC/ARQ systems using rate-compatible low-density parity-check codes," in *Proc. ITC*, 2000, vol. 48, no. 6, pp. 948–959.
[30] M. Ilyas, M. Kim, and H. Radha, "Reducing packet losses in networks of commodity IEEE 802.15.4 sensor motes using cooperative communication and diversity combination," in *Proc. IEEE INFOCOM*, 2009, pp. 1818–1826.

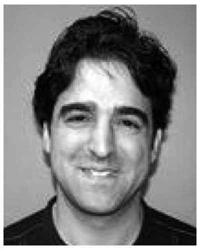
Chenxi Qiu received the B.S. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2009, and is currently pursuing the Ph.D. degree in electrical and computer engineering at Clemson University, Clemson, SC, USA.

His research interests include sensor networks and wireless networks.

Haiying Shen (M'06–SM'13) received the B.S. degree in computer science and engineering from Tongji University, Shanghai, China, in 2000, and the MS and Ph.D. degrees in computer engineering from Wayne State University, Detroit, MI, USA, in 2004 and 2006, respectively.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing.

Dr. Shen is a Microsoft Faculty Fellow of 2010 and a member of the Association for Computing Machinery (ACM). She was the Program Co-Chair for a number of international conferences and member of the Program Committee of many leading conferences.



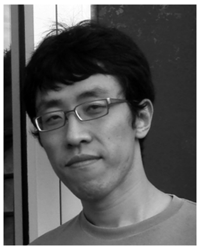
Sohraab Soltani received the B.E. degree (Hons.) from the Shiraz University, Shiraz, Iran, in 2003, and the M.S. and Ph.D. degrees from Michigan State University (MSU), East Lansing, MI, USA, in 2006 and 2009, respectively, all in computer science.

He has been a PI or Co-PI in several NSF, SBIR, and STTR projects. His research interests include computer security, stochastic models, data mining, statistical information inference, wireless video and cognitive mobile communications, ad hoc and sensor networks, and protocol design. He has extensive knowledge in computer security, anomaly and intrusion detection, protocol design, statistical data analysis, computer networking, and wireless multimedia communication and has published several papers in competitive conferences and journals in these areas.

Dr. Soltani received a prestigious dissertation completion award from MSU due to his outstanding research effort.

Karan Sapra received the B.S. degree in computer engineering from Clemson University, Clemson, SC, USA, in 2011, and is currently pursuing the Ph.D. degree in electrical and computer engineering at Clemson University.

His research interests include P2P networks, distributed and parallel computing systems, and cloud computing.



Hao Jiang received the B.S. and Master's degrees in computer science from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2006 and 2008, respectively, and the Ph.D. degree in computer science from Clemson University, Clemson, SC, USA, in 2012.

His research interests mainly focus on distributed algorithm design in embedded systems.

Jason O. Hallstrom (A'08–M'08) received both the B.S. degree in systems analysis and M.A. degree in economics from Miami University, Miami, FL, USA, in 1998, and both the M.S. and Ph.D. degrees in computer and information science from The Ohio State University, Columbus, OH, USA, in 2004.

He is currently an Associate Professor with the Computer Science Division, School of Computing, Clemson University, Clemson, SC, USA, and serves as the Deputy Director of the Institute of Computational Ecology. He directs the Dependable Systems Research Group (DSRG). Through the Institute of Computational Ecology, he serves as Technology Director for the Intelligent River program.