Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 4974165, 22 pages
https://doi.org/10.1155/2017/4974165

*Research Article*

# Efficient Network Coding with Interference-Awareness and Neighbor States Updating in Wireless Networks

**Xiaojiang Chen, Jingjing Zhao, Dan Xu, Shumin Cao, Haitao Li, Xianjia Meng, and Dingyi Fang**

*Northwest University, Xi'an, China*

Correspondence should be addressed to Xianjia Meng; xianjiam@nwu.edu.cn and Dingyi Fang; dyf@nwu.edu.cn

Received 11 March 2017; Accepted 2 July 2017; Published 12 September 2017

Academic Editor: Bernard Cousin



Network coding is emerging as a promising technique that can provide significant improvements in the throughput of Internet of Things (IoT). Previous network coding schemes focus on several nodes, regardless of the topology and communication range in the whole network. Consequently, these schemes are greedy. Namely, all opportunities of combinations of packets in these nodes are exploited. We demonstrate that there is still room for whole network throughput improvement for these greedy design principles. Thus, in this paper, we propose a novel network coding scheme, ECS (Efficient Coding Scheme), which is designed to achieve a higher throughput improvement with lower computational complexity and buffer occupancy compared to current greedy schemes for wireless mesh networks. ECS utilizes the knowledge of the topologies to minimize interference and obtain more throughput. We also prove that the widely used expected transmission count metric (ETX) in opportunistic listening has an inherent error ratio that would lead to decoding failure. ECS therefore exploits a more reliable broadcast protocol to decrease the impact of this inherent error ratio in ETX. Simulation results show that ECS can greatly improve the performance of network coding and decrease buffer occupancy.

## 1. Introduction

Have you ever felt exhausted for the bad network while you were just immersed in your computer games like LoL? Throughput improvement in wireless mesh networks is also a major problem for social networking and e-commerce platforms, where servers suffer dramatically heavy traffic and users are dying for fast and reliable network services. Many industries are looking to address this problem and improve throughput. Applications that involve high throughput, such as Ix [1] and Prophecy [2], provide high throughput performance with proxied or operating systems. These systems, however, call for hardware or system update and may be very costly. Moreover, it is unpractical for users to update system or hardware continuously. Network coding that emerged as a promising technique that can provide significant improvements with low cost, on the other hand, has attracted much interest. The notion of network coding allows relay nodes to not only forward but also process (i.e., XOR) the incoming packets. Despite all the scenarios mentioned before, network coding is also a great opportunity for IoT as many sensors can transmit information at the same time. But there is a main question one may ask: how many packets should be encoded (XOR) to get better improvements? Most current schemes, however, encode as many packets as possible to get maximized throughput at relays, which does not fully exploit network utility and throughput in the whole network. The whole network, however, would benefit from this bandwidth-efficient technique, with the same bandwidth but much higher throughput gains.

The foreground of potential throughput gain has received much attention since it was first proposed in [3]. It is subsequently found to be particularly suitable for wireless networks, since the broadcast nature allows the transmitting packets to be overheard by all its neighbors. Prior work focuses on addressing two problems: (i) throughput gain bounds with various scheduling schemes and theoretical analysis on encoding part [4–6] and (ii) transport protocols on decoding

part [7–9]. However, most of the network coding schemes like [10, 11], are under the assumption that the buffer and computing resource are sufficient, which is unlikely in most wireless mesh networks. Consequently, these schemes are greedy, exploiting all opportunities of broadcasting combinations of packets in a single transmission. However, with the fading and interference in wireless networks, the results of these greedy schemes are not always satisfying, which sometimes even shows a decrease of throughput [6]. Moreover, while there is much literature on the throughput improvement and bounds [4–6], it is still unclear for every relay to obtain better performance with proper encoding threshold when topology changes in the network. Here, we use encoding threshold to refer to the maximum number of native packets that can be encoded at a relay. This unclearness, however, is exacerbated in wireless mesh networks, where interference is more severe. That would leave the better throughput inaccessible.

This paper introduces ECS, an interference-aware network coding scheme in a robust manner which works in wireless mesh networks. In line with common practice in network coding schemes, ECS transfers encoded packets and overhears neighbors' packets for decoding. The challenge, however, is to make decisions about the encoding threshold of packets to encode once for every relay node (i.e., encoding number) to get more throughput gains with less interference and make full use of nodes in the whole network.

Unlike existing approaches, which consider buffer and computing resource as arbitrarily large, ECS exploits an interference-awareness model for relays in wireless networks, which maximizes network utility and throughput gains by reducing interference. Particularly, encoding too many packets at one node is unpractical with its limited resources (both buffer and computation). Moreover, due to the large number of combinations of native packets, relays have to forward the encoded packet to a large swath of receivers, whose neighbors have to keep in NAV timer to avoid collision with congestion control. Such nodes that keep in NAV timer, however, are a waste of resource, since even though they are not in the communication range of senders, they still cannot transmit any data for a certain period of time. Therefore, throughput can be improved if one can deal with the low utilization of nodes in NAV timer. That is to say, if we can make good use of nodes in the network, we can better meet the demand for throughput increment in wireless mesh networks.

To illustrate ECS's approach, Figure 1(a) shows a toy example with 11 nodes leveraging greedy network coding scheme, where the center node broadcasts an encoded packet of four native packets to the other four receiving nodes, which improves the throughput with less transmissions. It is also noticeable that all the silent nodes have to keep silent to avoid wireless interference, namely, waiting for the end of the transmission because of the occupancy of the wireless medium, even though they are not within communication range of the sending nodes. The large swaths of silent nodes are a waste of network resource; therefore throughput can be improved by "awaking" them. So can we "awake" maximum number of silent nodes to get more throughput gains? Can we get more throughput gains? The answer is yes. To do this, we transform these silent nodes to senders or receivers, while the sending coverage remains in a high ratio. Here, sending coverage refers to the ratio of sending nodes to all nodes in the whole network. Consider the scenario in Figure 1(b), where transmitters turn to be 3 and silent nodes in contrast turn to be 2, decreasing by 66.7%. The current throughput is 6, 1.25 times of that in Figure 1(a). Note that the sending coverage in Figure 1(a) is 9.09%, while the sending coverage in optimized solution shown in Figure 1(b) rises to 27.3%. In this way, we maximize the network capacity with minimum number of nodes in NAV timer, that is, minimum interference. In addition, receivers in the encoding scheme shown in Figure 1(b) store fewer packets in their buffer, reducing $2/4 = 50\%$. With less buffer occupancy than in Figure 1(a), the scheme in Figure 1(b) yields better results. That is to say, the seemingly best solution to improve throughput in Figure 1(a) indeed sacrifices the potential throughput for a maximum number of encoded packets delivered in a single transmission. Hence, the performance of greedy schemes alone, like the transmissions from Figure 1(a), can be improved by enlarging the sending coverage and decreasing the number of idle nodes. And that is why it can decrease possible interference while transmitting. An efficient networking coding scheme therefore needs to consider a main question: how to enlarge the sending coverage and decrease the number of idle nodes at the same time?

To do so, we divide nodes into three colors according to the three statuses: sending, receiving, and silent. By doing so, we simplify the problem into a graph coloring problem, a classical algorithm that deals with assignment. In contrast to the illustrative example in Figure 1(a), however, the evaluation of throughput in real-world wireless mesh networks, instead of a single transmission, is the total throughput in the whole network. Current schemes, however, are the theoretical bounds or are adaptive for certain given topologies, which are unlikely in wireless mesh networks. In fact, the changes of topologies are inevitable. In designing a scheme that can enlarge sending coverage as well as considering the changes of topology, we modify the objective function to minimize nodes in NAV timer and keep sending nodes in a high ratio, differentiating from the classical graph coloring. For every relay node, it "colors" nodes into as many senders as possible with certain limits to avoid collision and with the same strategy "colors" the neighbors of these senders to receivers. However, this is time-consuming and too complicated as the network scales up. Thus, to jump out of the local optimum, we combine the classical graph coloring algorithm with simulated annealing (SA) to "cool" the results. For every result in coloring, we iterate and discard the former result if the current result has better performance; otherwise, we keep it in a random rate. The end of iteration mainly depends on the initial temperature $T$ and current temperature $T'$, which are discussed in Section 5.4. After these procedures mentioned above, relays know the upper bound of encoding numbers, that is, refer to the number of their surrounding senders. Here, we call this upper bound of encoding numbers as encoding number threshold. Given these certain encoding number thresholds for every relay, the network encodes proper number of packets, which efficiently decrease interference and make good use of nodes. Therefore, the

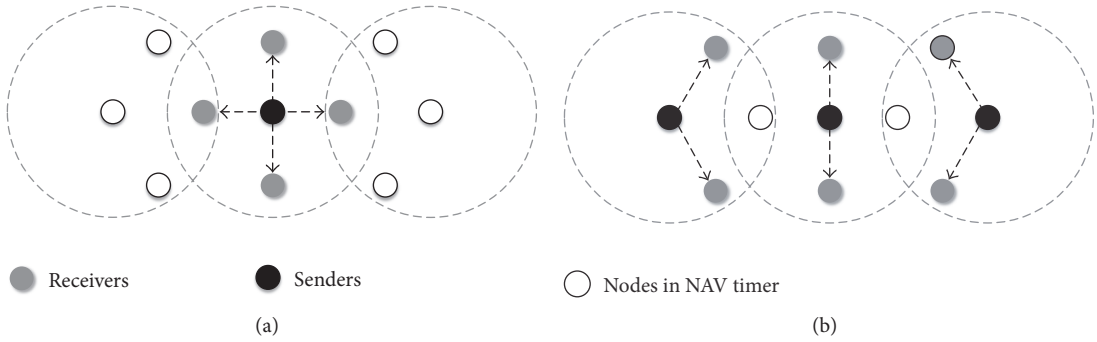



Figure 1: *Intuition underlying ECS's use of interference-aware model in coding packets*. (a) uses COPE as four packets encoded at the relay (black node), while (b) demonstrates our approach: two packets are, respectively, encoded at three relays, providing more throughput improvement.

whole network can obtain more throughput gains. We present the design in Section 5 and demonstrate in Section 5.3 the time complexity of the two.

Finally, while one could overhear packets and guess whether a neighbor has a particular packet leveraging opportunistic listening, network coding scheme today would require periodically computing ETX metric [12], which may make incorrect decisions in practice. Moreover, this is exacerbated when the network is of poor link quality. To solve this problem, relays can send request to get the information about what their neighbors store in the buffer. However, it is too costly to occupy the medium to exchange buffer information for each single transmission and therefore may reduce throughput. Hence, if we can find a low-cost approach to learn neighbor states and decrease the inherent error rate in ETX, the decoding procedure can be ensured. ECS carries decoding and encoding information while reserving the channel, namely, carries the total packets number encoded or buffered in RTS and CTS, which is in low cost and efficiently reduces the decoding failure.

Our simulations lead to the following findings:

(i) Receiving $\omega = [10, 300]$ packets per second, the throughput of ECS is 7 to 40 times higher than that of 802.11a/b/g and 3 times higher than that of greedy schemes, such as COPE scheme.

(ii) With same throughput gains, ECS stores fewer packets in the buffer than COPE, indicating that ECS has better efficiency and is of low cost.

*Our contribution of this paper includes the following*:

(i) This work presents a novel efficient scheme that scales whole network throughput by decreasing interference from overwhelmed encoding packets. To achieve this, we dynamically calculate an encoding number threshold for every relay node, according to communication range from topology. Finally, we show that our scheme can deliver throughput gains in mesh networks and any other networks that suffer great traffic.

(ii) We present a neighborhood listening state protocol, which reduces the unavoidable incorrectness when leveraging ETX by carrying coding information in RTS/CTS. Therefore, it decreases retransmissions and decoding failures in a low-cost approach.

## 2. Background

The attractiveness of network coding is marrying XOR and sending packets. Thus, it is a prerequisite for receivers to possess the ability of decoding the XOR-ed packets. To do so, receivers buffer packets that can be used to decode XOR-ed packets, which we call decoding resource. Differentiating the decoding resource, the network coding is classified into two categories. The first type of decoding resource only stores packets that have been sent by nodes, while the decoding resource of the second type includes the packets by opportunistic listening in addition.

The first category of network coding is demonstrated in Figure 2(a). In conventional cases, when Alice and Bob want to swap a pair of packets with the help of a router, the router has to forward the packets from two senders separately. Therefore, the number of transmissions needed is 4. After leveraging network coding, the router just needs to XOR the packets sent by Alice and Bob and broadcasts the XOR-ed packet. Alice and Bob can both hear the XOR-ed version and use their decoding resource to decode the XOR-ed packets. For example, Alice XORs the XOR-ed packet with her original sending packet; then she can get the message Bob wants to send to her. Consequently, the number of transmissions falls to 3.

The second category of network coding works in a more sophisticated network scenario. Take Figure 2(b) as an example. When these four persons want to exchange packets with network coding in a wheel topology, the senders not only send but also opportunistically listen to others. For example, let us assume that Alice can only have packet $p_1$ in its decoding resource in the first category, but it can also buffer packet $p_4$ and packet $p_2$ via overhearing to Sally and John. The relay then broadcasts packet $p$ ($p = p_1 \oplus p_2 \oplus p_3 \oplus p_4$) after the receivers (but also the senders in this scenario) have enough decoding resource, and the receivers can decode the packet.

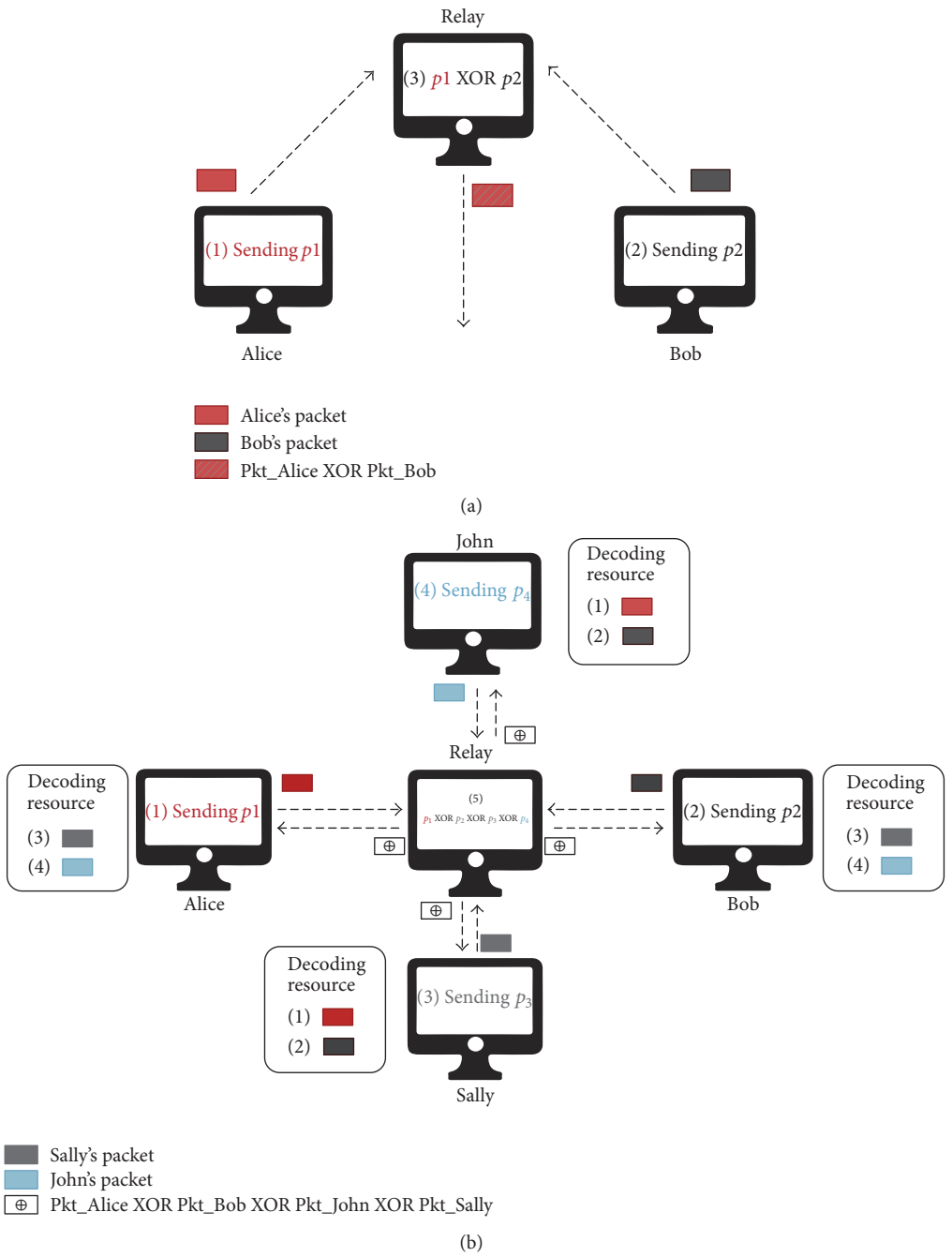



Figure 2: A brief illustration of network coding.

## 3. ECS Overview

ECS is an interference-aware scheme that can provide lower buffer occupancy and yield a higher throughput performance, much adaptive than most of the greedy network coding schemes including COPE [10]. To better improve the network throughout, ECS goes through the following steps:

(i) Depending on whether there are many encoding opportunities in the network, ECS first decides the start time of the network coding, using the technique in Section 4.

(ii) By modeling and giving solutions to the interference-aware model as described in Section 5, ECS calculates the maximum number of packets that a relay node can encode with the knowledge of topology.

(iii) Section 6 presents a neighbor status updating protocol. ECS enhances the accuracy on neighbor learning

scheme by adding encoding and decoding information in RTS and CTS while reserving the channel.

The next few sections elaborate on the above things, providing the technical details.

## 4. Encoding Time Decision

In order to deal with latency when there are few opportunities of encoding that can be exploited, we introduce encoding time decision, which turns on network coding at relays only when there are more encoding opportunities around them. Note that a key feature which arises in network coding is that when relays want to encode several packets in the transmission, they wait a period of time $\theta$ for the arrival of packets to encode, that is, encoding opportunities. In addition, all the receivers also have to wait for this encoding process and spend time decoding. In conclusion, the use of network coding increases latency in the network. This latency is inevitable in practice and is exacerbated in the cases where less opportunities of broadcasting combinations of packets can be exploited. The lack of opportunities leads to longer delays and provides fewer throughput gains. Hence, we need to find a tradeoff between latency and encoding opportunities and start to use network coding when the encoding opportunities exceed a certain threshold. Current schemes, however, leverage network coding all the time for throughput gains, which brings unwelcome latency especially when the throughput improvement is not tremendous. Our schemes, therefore, can efficiently reduce latency.

In order to decide the scenarios to start network coding, we first introduce two parameters that needs to be considered and then give some reasonable values after our simulations.

### 4.1. Definition

*Inverse Packets.* Let $p_1$ and $p_2$ be the packets relay $R$ receives sequently. $s_i$ and $n_i$ denote the previous hop and next hop of $p_i$, respectively. $t_i$ is the time when packet $p_i$ has been received by the relay $R$. The neighbors set of $m$ is denoted as $N(m)$. If $p_1, p_2$ are a pair of inverse packets of the relay $R$, then $n_1 \in N(s_2)$, $n_1 \notin N(s_1)$, $n_2 \in N(s_1)$, $n_2 \notin N(s_2)$, and $|t_1 - t_2| < \theta$. $\theta$ denotes the time period waiting for the arrival of inverse packets.

Take Figure 2(b) as an example. Let us assume that John wants to send the packet $p_1$ to Sally, while Sally also sends a packet $p_2$ to John. When both $p_1$ and $p_2$ arrive at the relay, the next hop of $p_1$ turns to Sally and the next hop for $p_2$ is John. Therefore, $n_1 = Sally$, $s_1 = John$, $n_2 = John$, $s_2 = Sally$. The neighbors set of John $N(s_1) = \{Alice, Bob\}$, and Sally has the same neighbors' set as John. Therefore, $p_2$ and $p_1$ satisfy the requirements: $n_1 \in N(s_2)$, $n_1 \notin N(s_1)$, $n_2 \in N(s_1)$, and $n_2 \notin N(s_2)$. $t_1$ and $t_2$ denote the times when packets $p_1$ and $p_2$ have been received by the relay $R$, respectively. If $t_1$ and $t_2$ satisfy $|t_1 - t_2| < \theta$, then $p_1$ and $p_2$ are inverse packets.

*Data Flow Complexity (DFC).* If a relay receives $r$ native packets that contain $i$ pairs of inverse packets in a unit of time, then DFC $= 2i/r$.

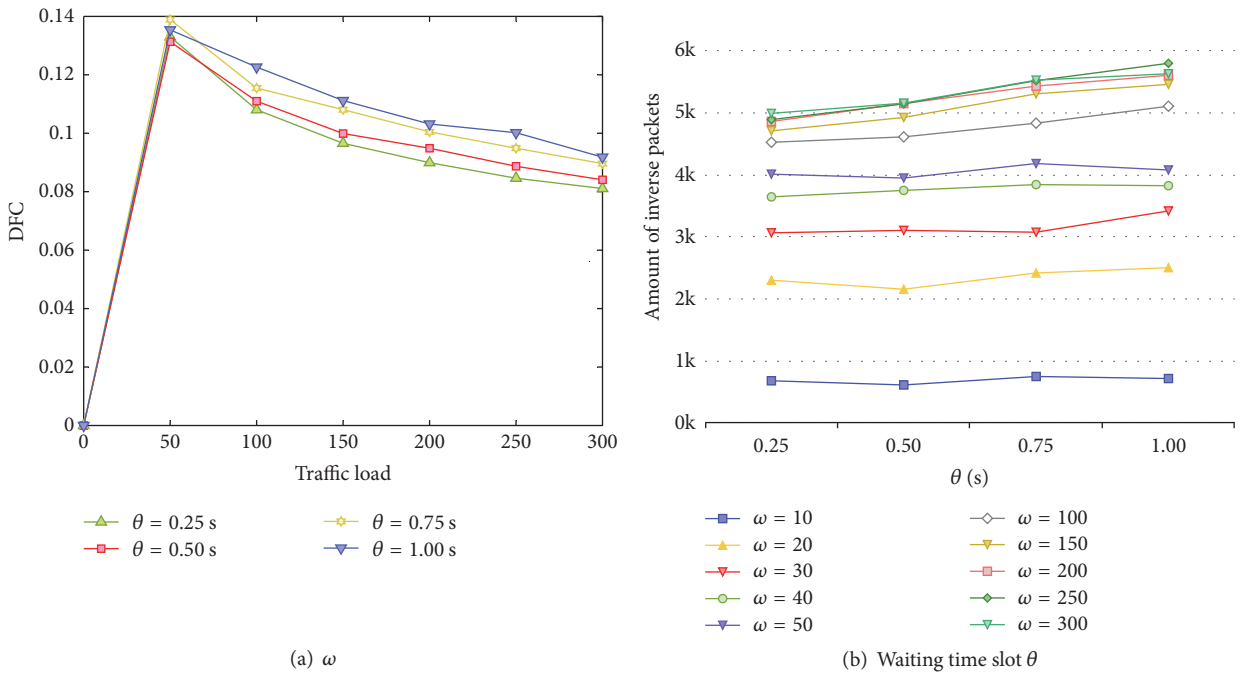
In summary, inverse packets are a pair of packets which can be encoded at a relay and decoded at receivers. The ratio of inverse packets to native packets, DFC, can therefore denote the encoding opportunities regardless of load. As shown in the definition, the calculation of DFC does not rely on topology or other network status; rather, it can be applied in any networks separately. The nodes would not start using network coding unless DFC exceeds a certain threshold $\zeta$. Starting time decision is just like a sensor in voice-activated light in hall, which only turns on when it is noisy in the hall. As opposed to listening to the noise around, however, ECS exploits the DFC to identify the data complexity and the right time to use network coding.

*4.2. Calibrating Parameters.* In this part, we calibrate parameters $\zeta$ and $\theta$ through numerous simulations. It should be noted that the related parameters can be modified with various users' need. One important setting of our simulation is that we select sender and receiver randomly, and nodes are randomly distributed. Thus, we set up an ad hoc network with randomly distributed nodes.

*4.2.1. Open Time Threshold $\zeta$.* With the same setup detailed in Section 7, we turn off network coding in the simulation and let the nodes send packets according to the set of $\omega$ (the amount of sending packets per second) and count the values of DFC according to $\theta$ with the 802.11 protocol. In the simulation $\omega = [10, 20, 30, 40, 50, \ldots, 300]$, there are 30 kinds of values in total, and in $\theta = [0.25, 0.5, 0.75, 1.00]$, there are 4 kinds of values in total. Therefore, every group of simulations (120 groups in all) repeats 1000 times. The results of simulation are shown in Figure 3.

Among the factors that can affect the DFC, $\theta$ in inverse packets definition and parameter $\omega$ are the most critical parameters, due to controlling network load. The impact of network traffic load ($\omega$) on data flow complexity (DFC) is depicted in Figure 3(a). For small load, nodes are more likely to forward packets in the forwarding queue to the next hop. However, as few packets are stored in the forward queues, new arrival packets are hard to be matched into inverse packets, and that is why DFC equals 0.07 when $\omega = 10$. As the load increases ($\omega < 30$), the number of packets in forwarding queues increases, leading to a higher DFC rate and a larger amount of inverse packets. When the number of packets in forwarding queues exceeds a certain threshold, DFC levels off, reflecting the fact that inverse packets have reached their capacity and cannot combine additional inverse packets. Additionally, with the variation of $\theta$, DFC shows only slight difference as load ($\omega$) goes up, which reveals that $\omega$ has a greater impact on DFC than $\theta$.

Figure 3(b) plots the amount of inverse packets with a group of $\omega$ when $\theta$ goes up. When $\omega$ is low, the change of the number of inverse packets is very slight. Hence, the impact of $\theta$ on inverse packets is correspondingly modest. As $\omega$ increases, forwarding queues have more packets due to the high load; thus the amount of inverse packets goes up. Surprisingly, these curves are very close, reflecting the fact that network has reached its capacity and cannot sustain additional load.




(a) $\omega$ (b) Waiting time slot $\theta$

FIGURE 3: The number of sending packets $\omega$ and waiting time slot $\theta$ versus DFC.

The above simulation results reveal that when $\omega > 10$ (unit: packets per second), DFC is always beyond 0.07. From Figure 3(b), when $\omega = 10$, the amount of inverse packets is scarce, fewer than 1000 per minute. In these scenarios, packets can easily access the medium to transfer data due to the low load. The coding opportunities are also scarce due to the sparse packets in the forwarding queue. Thus, we suppose that very few throughput gains can be obtained with ECS in these scenarios. In summary, we take $\zeta = 0.07$. That is to say, if and only if $\zeta > 0.07$, the coding procedure will be started.

*4.2.2. Waiting Period $\theta$.* In this part, we discuss $\theta$, the waiting period for the arrival of inverse packets. Recall from inverse packets' definition that $\theta$ is the time waiting for inverse packets. We also set $\theta$ as the initial value for timer to overhear packets. In a same network scenario, the more time node waits for the inverse packets, the more packets it overhears and stores in its buffer. Thus we need to consider the buffer occupancy of nodes in the whole network, that is, how many packets nodes have overheard.

We start our flows again according to a Poisson process with ECS, that is, pick sender and receiver randomly and allocate certain memory (1 MB) for every node to store decoding resource. There are extra 10 MB for spare memory. That is to say, nodes will store decoding resource in spare memory after consuming 1 MB. The resource will be discarded if both memories run out. We classify nodes into three categories according to the usage of memory: overloaded nodes for those that use more than 1 MB memory; low-use nodes referring to those nodes whose memory usage is less than 0.25 MB; and the overhead of healthy nodes which is between 0.25 MB and 1 MB.

We repeat the simulation in varying load $\omega$ with ECS enabled. Figure 4 plots the distribution of the ratio of these three kinds of nodes. Figure 4(a) shows that the majority of nodes are unable to utilize the memory efficiently when the load is low, and the ratio remains flat with increasing $\theta$. This illuminates weak correlation between $\theta$ and memory usage. Note that only those on the network trunk intersection fully utilize the memory. As the load increases, some low-use nodes turn into healthy nodes, while the memory usage of some healthy nodes ramps up into overloaded nodes. These transactions are also related to the location of nodes. For those healthy nodes in the trunk, they turn to overloaded nodes and then the low-use nodes in subtrunk become healthy nodes. The low-use nodes in the margin, however, are still in low use due to the lack of flow. As a result, nodes in low use decrease gradually, while overloaded nodes leap. But it should be emphasized that this does not mean that waiting period is related to the topology because the topology does not decide how many packets nodes send. As load surges even higher, nodes in low use decrease, almost equal to the amount of healthy nodes. Surprisingly, overloaded nodes decrease, reflecting that low-use nodes in relative margin networks turn into healthy nodes as load in subtrunk network goes up.

The reason that we make decisions on $\theta$ is to increase the ratio of healthy nodes and decrease the overloaded nodes, most obviously shown in Figure 4. As a result, we set $\theta = 0.3$ in our later simulation. Note that load also affects the number of these three categories nodes, which, however, is unpractical to control with the setting. Thus we use $\theta$ to control the ratio of these three kinds of nodes. According to this observation, the ratio of overloaded nodes is below 5%, while the number of healthy nodes is increasing when $\theta = 0.3$. Thus $\theta$ is set to be 0.3.

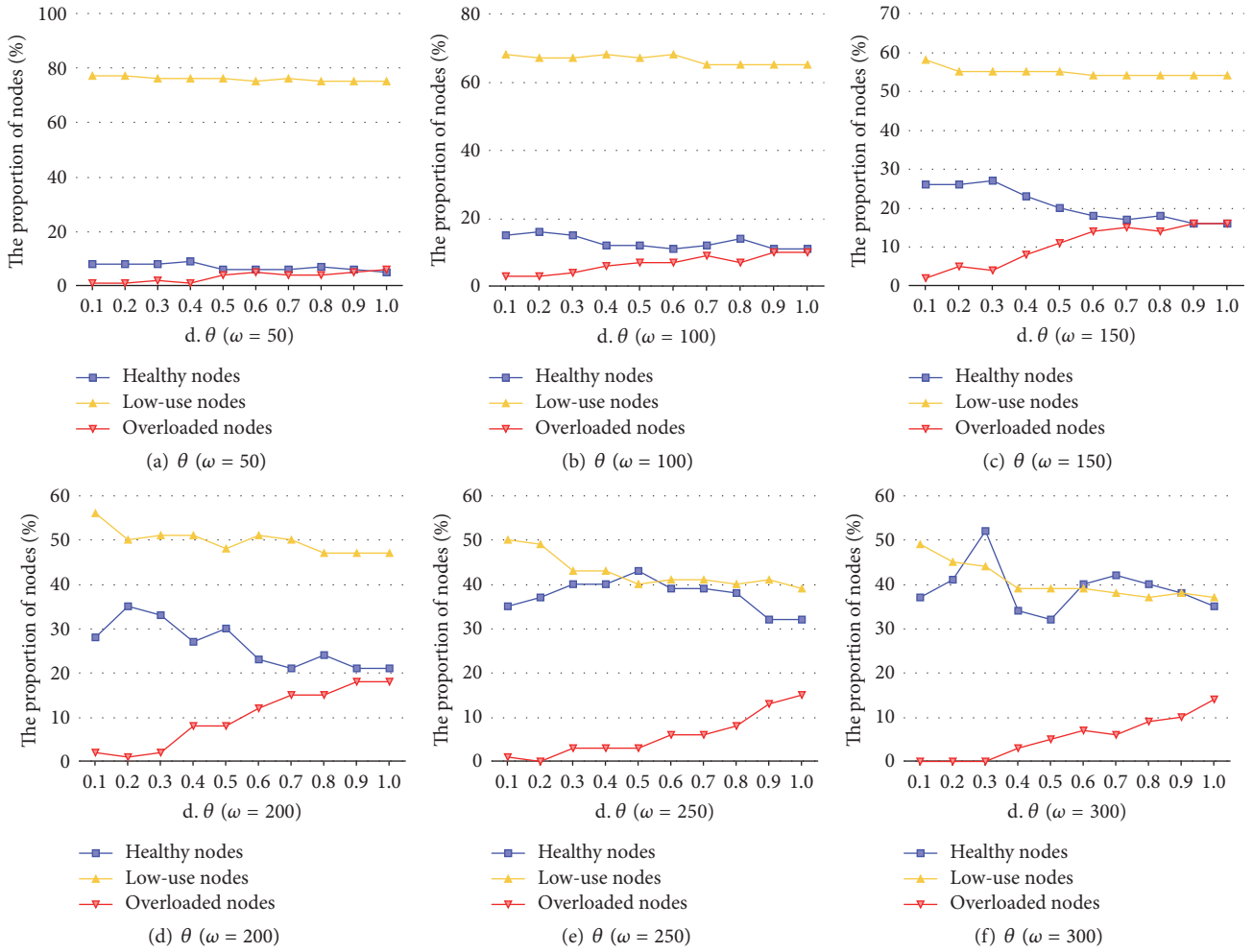



FIGURE 4: The proportion of all states (healthy, low-use, and overloaded) of nodes in various time periods $\theta$.

## 5. Encoding Number Threshold Model

After acquiring the start point of coding scheme, ECS figures out the number of packets needed to be coded together in a single transmission to get more throughput gains. Many greedy schemes including COPE do not constrain this number, pursuing maximum throughput in a single transmission, which would encounter encoding and decoding failure and, most importantly, may reduce throughput when applied in the whole network. Exploiting not enough combinations of packets to broadcast, on the contrary, leaves much space for throughput gains due to the potential opportunities. ECS therefore needs to find a tradeoff between throughput gains and encoding numbers. Current schemes, however, are adaptive only for several general and unchanging topologies, which is unlikely in wireless mesh networks. In fact, the change of topology is inevitable. By maximizing utilization ratio every time when the topology changes to obtain throughput gains in the whole network, ECS makes use of topology and communication range, which yields a better result.

In order to get the encoding number threshold of packets, we divide nodes into three colors according to the three statuses: sending, receiving, and silent. By doing so, we simplify the problem into graph coloring, a classical algorithm to deal with assignment. Rather than coloring neighbor nodes with different colors to get a minimum number of colors in traditional coloring problem, we modify the objective function to maximize the difference between the ratio of total transmitters and the ratio of silent nodes to get optimal throughput gains in wireless mesh networks. Then we leverage SA to "cool" the results. In designing a scheme to jump out the local optimal results, we combine the classical graph coloring algorithm with simulated annealing (SA).

*5.1. Formulation.* As mentioned before, ECS aims at high usage rate and sending coverage. Therefore, the scheme is simplified into a kind of problem of minimizing nodes in NAV timer (silent nodes) while keeping sending nodes in a fairly high rate, which belongs to a particular channel assignment.

Coloring algorithm in connected graphs and conflict graphs works well for such channel assignment problems. Instead of coloring neighbors with a minimum number of different colors, the evaluation functions in ECS are depicted

as follows. Let an undirected graph $G = (V, E)$ represent a wireless mesh network: $|V| = n$ and $|E| = m$. Node $v_i$ is the node in the network, while $e_{ij}$ is the single-frequency link. The optional status for each node is in $C = \{1, 2, 3\}$ where 1 is for sending, 2 is for receiving, and 3 is for silent. Let $x_{ic} = 1$ be true to color node $i$ with color $c$ and 0 for not coloring with color $c$. And the model is shown as follows:

$$\begin{aligned}
\max \quad & f(x) = \frac{\left(\sum_{i=1}^{n} x_{i1} - \sum_{i=1}^{n} x_{i3}\right)}{n} \\
\text{s.t.} \quad & x_{ic} \in \{0, 1\}, \quad 1 \le i \le n, \ c \in C \\
& \sum_{k \in C} x_{ik} = 1, \quad i = 1, 2, 3, \ldots, n \\
& \sum_{e_{ij} \in E} x_{j1} = 0, \\
& x_{i2} = 1 \\
& \sum_{e_{ij} \in E} x_{j2} \in \{1\} \cup \{2n \mid n \in N_+\}, \\
& x_{i1} = 1.
\end{aligned} \tag{1}$$

*5.2. Design Specification.* In finding solution to this problem, we are inspired by simulated annealing algorithm. Simulated annealing algorithm is a combination of random algorithm and greedy algorithm. Greedy algorithm can get a swift solution but may easily be stuck in local optimum. Simulated annealing algorithm, on the contrary, can approach global optimal solution by discarding local optimal solutions with a certain probability.

Let $T$ (sufficiently large) be the original temperature of annealing algorithm. The end of temperature of annealing algorithm is $T'$ ($T' > 0$) and annealing rate is $r$ ($0 < r < 1$). The evaluation function of annealing algorithm is the objective function in Section 5.1, whose answer is given by initial packet coding algorithm. Let $f_i$ be the current local optimum of simulated annealing algorithm and let $f_{i-1}$ be the previous one. The temperature increment is $\Delta t = f_{i-1}(x) - f_i(x)$. If $\Delta t \le 0$, accept $f_i$ as the final result; otherwise, calculate $p = \exp(-\Delta t/T)$ and accept the result with the probability of $p$. If $T \le T'$, finish the algorithm while cooling the result with $T = rT$ for next round of iteration. The whole procedure that simulated annealing algorithm takes is illustrated in Algorithm 1.

The challenge in this procedure is determining the values of $T$, $T'$, and $r$, which control the speed of this algorithm. The result we get at last will be far away from the optimal solution if the speed is too fast and will be a waste of resource on the contrary. Generally speaking, it is certain for $r$ (e.g., 0.95) and alternative for $T$ and $T'$ which are discussed in detail in Section 5.4.

But what is the answer to $f(x)$? With the absence of $f(x)$, it is impossible to get the optimal solution. As mentioned before, we use coloring algorithm to design this greedy strategy. Typically, we choose the degree of vertexes considering the special features of network coding.

```
(1) FinalResult = f_0(x);
(2) while (T > T') do
(3)     Δt = FinalResult − f_i(x);
(4)     if Δt ≤ 0 then
(5)         FinalResult = f_i(x);
(6)     else
(7)         if random(0, 1) < exp(−Δt/T) then
(8)             FinalResult = f_i(x);
(9)         end if
(10)    end if
(11)    T = rT;
(12)    i + +;
(13) end while
```

Algorithm 1: Simulated annealing procedure.

Let $d_G(V_i)$ be the degree of vertex $i$ in the undirected graph $G$. The set of neighborhood of $i$ is $s_i$; that is, $s_i = \{v_j \mid e_{ij} \in E\}$. $W_{ic}$ represents the set of neighbor of $i$ that is colored in color $c$; that is, $W_{ic} = \{v_j \mid v_j \in S_i, \ x_{jc} = 1, \ c \in C\}$. The sum of the degree of $i$'s neighbor is $d'_G(v_i)$; that is, $d'_G(v_i) = \sum d_G(v_j)$, $v_j \in S_i$.

Nodes execute the procedure as follows.

*Step 1* (paint all the nodes into silent and put all of them in a temporary set $B$). Color the current node (the running node, i.e., *this* in Algorithm 2) and search for two nonadjacent neighbors of *this* and color them into receiving status. If these two neighbors do not exist, color the node that owns largest $d'_G$ among neighbors of *this* into receiving status.

*Step 2* (find sending nodes as many as possible). The node $v_i$ cannot be painted to sending unless all the neighbors of $v_i$ are silent. After that, find the node $v_j$ with largest $d'_G$, and color $v_i$ to receiving status. Then exclude the nodes that can only unicast with $v_i$. Find neighbors $v_k$ of $v_i$ which meet the following requirements: (a) $v_k$ is in silent, (b) $v_k$ is not adjacent to $v_j$, and (c) $v_k$ has only one sending node $v_i$ in its neighbor set. Receiving node $v_k$ has the largest $d'_G$. If $v_k$ does not exist, $v_i$ and $v_j$ can only unicast.

*Step 3* (increase the broadcast amount of sending nodes). Clean temporary $B$ and put all the sending nodes in it. Scan all the nodes in $B$; if the neighbor of node $v_i$ has only one receiving node or the silent nodes are not more than 2, remove $v_i$ from $B$; otherwise, find two neighbors $v_j$ and $v_k$ of $v_i$ which meet the following requirements: (a) they are not adjacent to each other and (b) they both have and only have one sending node (i.e., $v_i$). If $v_j$ and $v_k$ do not exist, remove $v_i$; otherwise, color both nodes ($v_j$ and $v_k$) into receiving status. End Step 3 until the set $B$ is empty.

*5.3. Computational Complexity Analysis.* The key to network coding is the scheme for different purposes. However, some schemes are very sophisticated, as [13–16] are NP-complete. COPE has a much lower complexity, that is, $O(n^2)$. The

(1) $B = V$; $i = this$; $x_{i1} = 1$; $x_{i3} = 0$;
(2) **if** $\exists v_j, v_k\ (v_j v_k \in S_i, v_j \notin S_k)$ **then**
(3) $\quad x_{j2} = 1$; $x_{j3} = 0$; $x_{k2} = 1$; $x_{k3} = 0$;
(4) **else**
(5) $\quad x_{j2} = 1$; $x_{j3} = 0$; $(d'_G(v_i) = \max(d'_G(S_i)))$;
(6) **end if**
(7) **while** $B \neq \emptyset$ **do**
(8) $\quad$ Pick $v_i$ from $B$ randomly;
(9) $\quad$ **if** $sizeof(W_{i1} + W_{i2}) \neq 0 \parallel sizeof(W_{i3}) = 0 \parallel sizeof(S_i) = 0$ **then**
(10) $\quad\quad B = B - \{V_i\}$; **continue**
(11) $\quad$ **end if**
(12) $\quad$ **if** $\exists v_j\ (v_j \in S_i, d'_G(v_j) = \max(d'_G(S_i)), W_{j1} = 0)$ **then**
(13) $\quad\quad x_{i1} = 1$; $x_{i3} = 0$; $x_{j1} = 0$; $x_{j3} = 0$;
(14) $\quad\quad$ **if** $\exists v_k\ (v_k \in S_i, v_k \notin S_j, x_{k3} = 1, d'_G(v_k) = \max(d'_G(\{v_{kn}\})), w_{k1} = 1)$ **then**
(15) $\quad\quad\quad x_{k2} = 1$; $x_{k3} = 0$
(16) $\quad\quad$ **end if**
(17) $\quad$ **end if**
(18) **end while**
(19) $B = \{v_i \mid v_{i1} = 1\}$
(20) **while** $B \neq \emptyset$ **do**
(21) $\quad$ Pick $v_i$ from $B$ where $d_G(v_j) = \max(d_G(B))$
(22) $\quad$ **if** $sizeof(W_{i2}) = 1 \parallel sizeof(W_{i3}) < 2$ **then**
(23) $\quad\quad B = B - \{v_i\}$; **continue**;
(24) $\quad$ **end if**
(25) $\quad$ **if** $\exists v_j, v_k\ (v_j \notin S_k, v_j \in S_p, v_k \in S_p, v_p \in W_{i2}, x_{j_3} = 1, x_{k3} = 1)$ **then**
(26) $\quad\quad x_{j_2} = 1$; $x_{j_3} = 0$; $x_{k2} = 1$; $x_{k3} = 0$
(27) $\quad$ **else**
(28) $\quad\quad B = B - \{v_i\}$
(29) $\quad$ **end if**
(30) **end while**
(31) **return** $\sum x_{i1} - \sum x_{i3}/sizeof(V)$

Algorithm 2: Initial packet coding coloring.

complexity of our scheme, on the contrary, is much lower with $O(2n)$, which is more practical.

*5.4. $T$ and $T'$.* We repeat the large scale simulation with different coloring decision, that is, unicast coloring, coloring with COPE, and coloring with ECS. All of the coloring decisions follow the same algorithm framework, and it is the solution to $f(x)$ that varies. Note that each node can only communicate with at most one neighbor once in unicast coloring and it can multicast to as many as reachable neighbors with greedy algorithm in COPE. In ECS, nodes make color decisions with the algorithm proposed in Section 5. The flows again arrive according to a Poisson process and pick sender and receiver randomly with a radio range of 175 m and the density of 1/194 $m^2$.

The number of nodes involved in communication shown in Figure 5 is much larger than unicast, revealing the advantages of network coding. However, the sending coverage of unicast is wider than COPE due to the MAC protocol based on CSMA/CA. ECS holds a more wider coverage than COPE and a slightly smaller number of communication nodes than COPE, which is obviously bigger than unicast.

Figure 5 plots the summary results in a bar graph. The sending ratio (the number of sending nodes/number of nodes) of unicast is beyond 13%. However, the utilization ratio ((sending nodes + receiving nodes)/nodes) is low, around 26%. COPE provides 50% improvements in utilization ratio. However, the sending ratio is lower than 9%. ECS holds approximate sending ratio with COPE, while maintaining a high sending coverage ratio. From the perspective of *this*, COPE constrains the encoding number: it broadcasts an encoding packet of four. ECS lowers this number to 2, avoiding affecting other nodes.

The next question is to determine $T$ and $T'$. From the definition of the model, we can draw a hidden condition:

$$\sum_{i=1}^{n} \frac{x_{ip}}{n} + \sum_{i=1}^{n} \frac{x_{iq}}{n} + \sum_{i=1}^{n} \frac{x_{ir}}{n} = 1, \quad (p = 1,\ q = 2,\ r = 3). \tag{2}$$

Receiving nodes as the target nodes of communication apparently has a correlation with sending nodes in number. Temporarily let

$$\sum_{i=1}^{n} x_{iq} = c \sum_{i=1}^{n} x_{ip}, \quad (p = 1,\ q = 2), \tag{3}$$

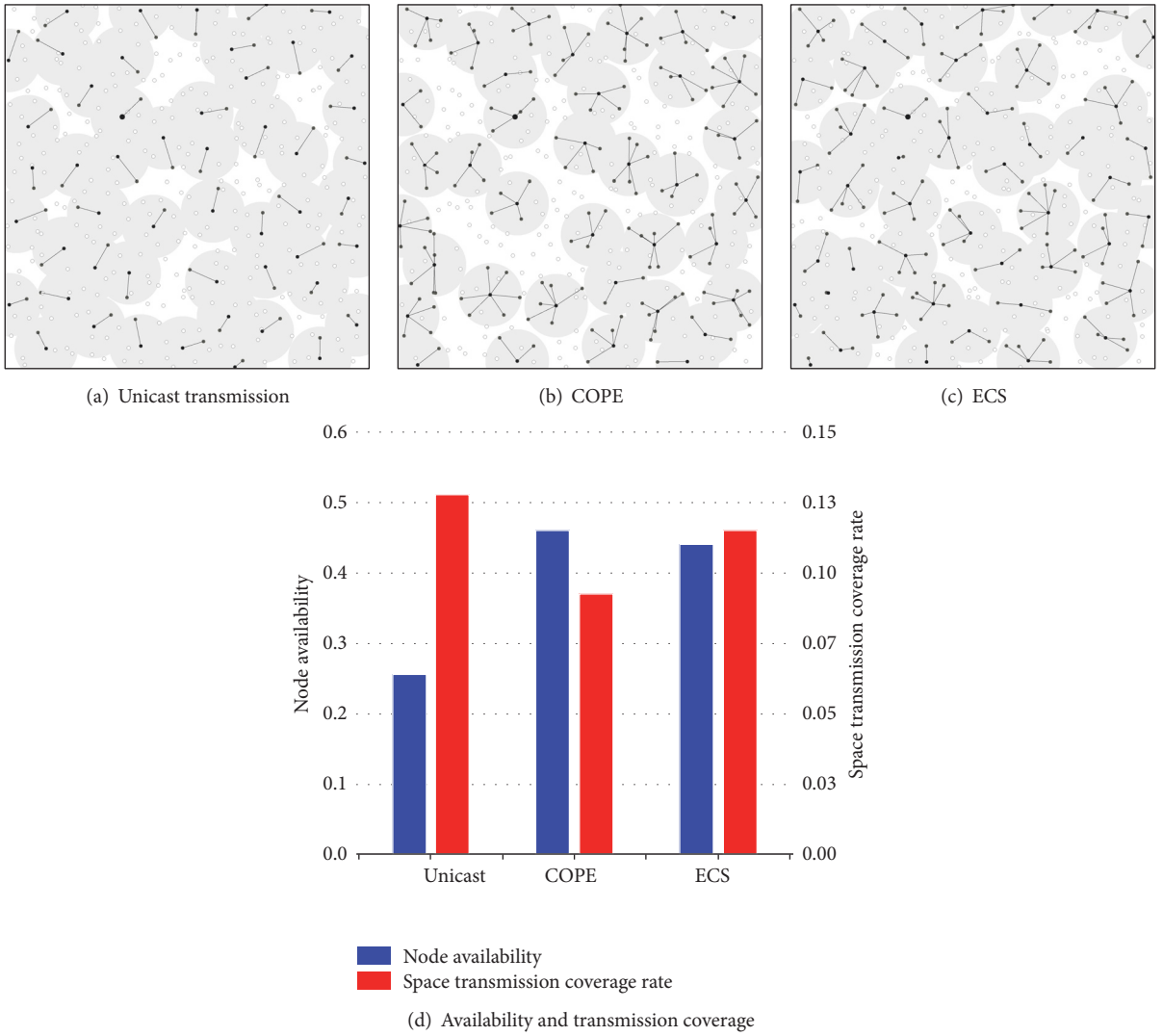



FIGURE 5: *The result of coloring*. Sending nodes are depicted in black, and receiving nodes are in light gray. White nodes are for nodes in NAV timer and the sending coverage is in light gray regions. The bigger black node is the algorithm execution node *this*.

where $c$ is the constant coefficient affected by the density of deployment and $c > 1$, so (3) can be transformed into

$$(1+c)\sum_{i=1}^{n}\frac{x_{ip}}{n}+\sum_{i=1}^{n}\frac{x_{iq}}{n}=1,\quad (p=1,\ q=3). \tag{4}$$

Substituting (4) into the objective function, we can get

$$\max\quad f(x)=\frac{1}{1+c}-\frac{2+c}{n(1+c)}\sum_{i=1}^{n}x_{iq},\quad (q=3) \tag{5}$$

due to

$$\frac{2+c}{n(1+c)}\sum_{i=1}^{n}x_{iq}>0 \tag{6}$$

which is fixed, so the objective function is equivalent to

$$\min\quad g(x)=\sum_{i=1}^{n}\frac{x_{iq}}{n},\quad (q=3). \tag{7}$$

$G(x)$ and $f(x)$ are in the same order of magnitude, and both of them are influenced by $T$ and $T'$. We simplify the objective function in this way.

We take $T \in [0.0.1, 1]$ of jump 0.01, $T' \in [0.0001, 0.01]$ of jump 0.0001, and 10000 groups of coloring simulations in total. For each $T$ and $T'$, we make color decisions for 100 times repeatedly and take the mean of coloring results $g(x)$ as a final result of this group simulation. Meanwhile, we count the time (unit: ms) of coloring 100 times as the total running time for this group simulation. Figure 6 plots simulation results. The effects of equivalent conditions $g(x)$ for $T$ and $T'$ are depicted in Figure 6(a), and the effect of $T$ and $T'$ impact on coloring duration is illuminated in Figure 6(b). Figures 6(c) and 6(d) plot the equivalent distribution figure of the two effects.

From Figures 6(c) and 6(d), $T = 1$ is a big enough value related to $g(x)$, and $T'$ has greater impact on $g(x)$. When $T' < 0.001$, $g(x)$ is nearly close to the minimum. In addition, the common feature of the running time influenced by $T$

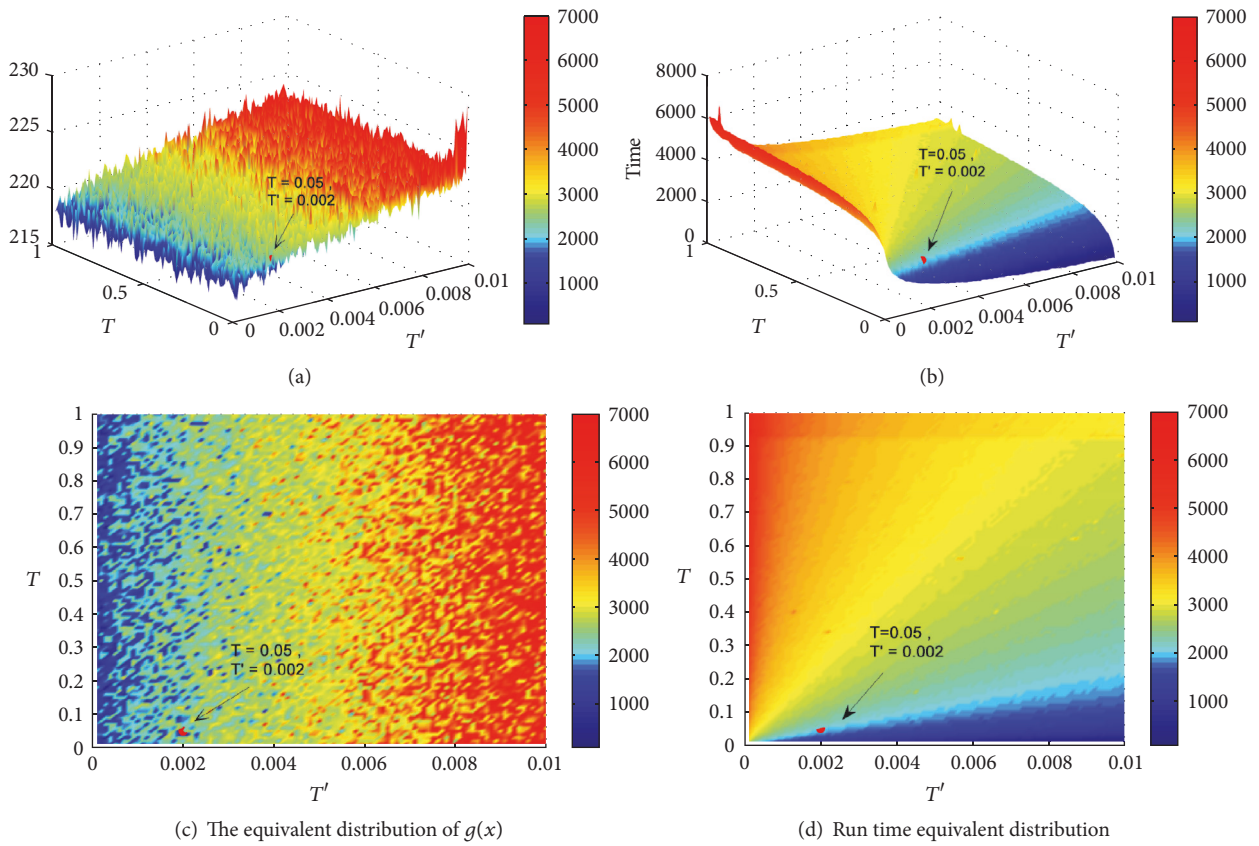



FIGURE 6: *The distribution in various $T$ and $T'$*. (a) demonstrates how $T$ and $T'$ influence $g(x)$. (b) shows the relationship between these two parameters and the cost of the whole coloring algorithm.

and $T'$ is apparent. Taking $g(x)$ and the running time into consideration, we finally take $T = 0.05$ and $T' = 0.002$ as the cooling speed.

## 6. Broadcast and Retransmissions

After making decisions about the encoding threshold, we need to reduce retransmissions due to the decoding failure. In this section, we describe how ECS learns neighbor state by altering the broadcast protocol.

Decoding failure is more likely to happen when the relay nodes make wrong guesses with opportunistic listening. In particular, we demonstrate in Appendix that there remains an inherent error ratio when leveraging ETX in opportunistic listening. Consequently, relay nodes have to reencode packets or send the packets one by one, which surely reduce the throughput. Transmitting control information alone is simple and reliable; however, it is too costly. Thus, it is crucial to find an approach to learn neighbors' states in a more reliable and low-cost way.

Our solution is to alter the broadcast protocol, which piggybacks on 802.11 broadcast and benefits from its control frame. Figure 7 shows the format of these three control frames in ECS. To build a neighbor state update protocol, we need to make a few designs. On the sending side, whenever the sender gets an opportunity to send, it adds packet information to the RTS frame. After receiving the RTS frame, the receiver checks the packet information in the received RTS frame and sends the CTS frame which adds its capability of decoding to the sender. When the CTS frame arrives at the sender, the node extracts the decoding capability in the CTS frame. Further procedure depends on whether the receiver can decode the encoded packet. If the receiver cannot decode the packet, senders need to reencode with appropriate packets; that is, the secondary encoding packet is the subset of the first packet. Then, the sender sends the RTS frame to the receiver and transmits the data frame. After successfully decoding the data with the reencoded packet, the receiver acks and ends this handshake. We add the times of handshake in CSMA/CA protocol. Therefore we can discard the nodes that cannot decode at receiver through the handshaking phase to decrease decoding failure.

*6.1. Updating Neighbor State.* Learning neighbor state is a major issue for network coding, which can be intelligently guessed by ETX metric. However, we prove in Appendix that this guess has inherent error rate. If the guess fails, the coded packet forwarded by transmit node would be undecodable by some next hop. Therefore, the transmit node needs to check the neighbors' ack and retransmit the undecodable packets. Such an approach works; however, it may decrease throughput in the whole network.

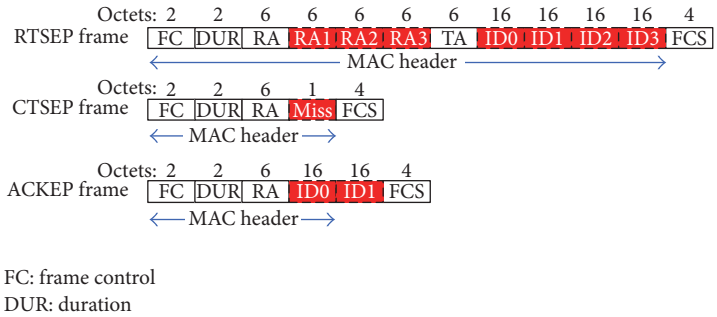



Figure 7: *The format of control frame*. The figure shows the format of control frame in 802.11 protocol and ECS. The field with white background color is what 802.11 protocol has presented. The highlighted fields are the added parts in ECS (e.g., as four encoded packets).

Table 1: Valid type and subtype combinations.

| Type value (bit 3 \| bit 2) | Type description | Subtype value (bit 7 \| bit 4) | Subtype description |
|---|---|---|---|
| 01 | Control | 0001 | RTSEP (the set of 2) |
| 01 | Control | 0010 | RTSEP (the set of 4) |
| 01 | Control | 0011 | CTSEP |
| 01 | Control | 0101 | ACKEP |

To visually understand the problem, let us consider a simple communication shown in Figure 9. For example, sender wants to communicate a packet to Receiver 1 and Receiver 2. And all their neighbors should stay in NAV state to avoid interference. If Receiver 2 cannot decode the packet, its neighbors (except the senders) may waste a single NAV cycle. We therefore design a protocol that alters the control frame in broadcast protocol to avoid such waste.

*6.2. Control Frame Detail.* In this section, we will introduce how ECS extends the control frame in CSMA/CA protocol to minimize the cost in transmitting data.

The first problem is to uniquely identify a packet in the whole network. Inspired by Message Digest Algorithm (MD5), we use the digest (16 B) of each data frame to be its ID number, which keeps the same data frame format in CSMA/CA protocol. Thus, our system could be compatible with the protocols in other layers. However, to reduce the query delay, nodes need to save the computed ID number of the packets when they are putting this packet into buffer.

As each packet can be identified in the whole network, the next to consider is how we use the packets' ID to update the neighbor state. The set of receivers and the encoded information are demonstrated in the first RTS frame sent by senders; that is, the RTS frame contains field RA and data frame ID. We define this control frame as RTSEP (Request to Send Encoding Packet). Figure 7 shows the RTSEP frame format of broadcasting four encoded packets. The receiver can compute the encoded packets number according to the Type and Subtype fields in the RTSEP frame. We enrich the Type and Subtype fields based on 802.11 protocol, described in Table 1. After the receiver obtains the RTSEP frame, it queries the index of itself in the receiver collection. According to the index, it can ensure the slot of sending CTS frame. The receiver should check the packets in its buffer based on the IDs in the RTSEP frame. If only one packet could not be found, that is to say, the presending packets can be decoded, then the receiver sends the CTS frame to the sender, which includes the index of those unfounded packets. We call this CTS frame CTSEP (Clear to Send Encoding Package). The format of CTSEP frame is illustrated in Figure 7. The Miss field is 1 byte in length destined for the result of querying ID number. Moreover, when neighbors of the receiver obtain the CTSEP frame, they check the Miss field in the control frame. If the Miss field has more than one bit which are set to value 1 in data type, the neighbor nodes regress from NAV state.

After the sender receives all the CTSEP frame, it reencodes the packets with the knowledge of CTSEP frame and removes the nodes that cannot decode the encoded packet. For example, Receiver 1 and Receiver 2 can encode the packet, while Receiver 3 and Receiver 4 cannot; that is, Receiver 3 and Receiver 4 are removed from the set of receivers. The sender then resends the RTSEP frame that updates the encoding packets information and reports the change of receiver collection to its neighbor nodes. After waiting for a SIFS slot, the sender sends the encoding data frame. Figure 8 visually shows the above process, and Receiver 3 and Receiver 4 are removed in the second handshake.

*6.3. Control Frame Cost Analysis.* Now we formally define a control frame cost as $c$, which is the total length of control frame in one communication. Let $c_n$ be the cost of control frame through transmission by $n$ set of packets: $n = 1, \ldots, k$. In particular, $c_1$ is the control frame cost of unicast. Then the control frame cost benefit is computed as

$$g_n = \frac{nc_1 - c_n}{nc_1}. \tag{8}$$

The duration of unicast is equal to one RTS frame (20 B per frame) plus one CTS frame (14 B per frame) and one ACK

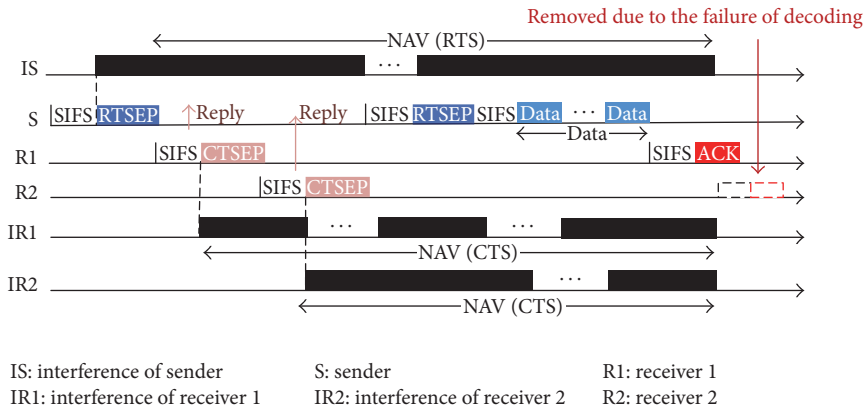



Figure 8: ECS broadcast protocol format.

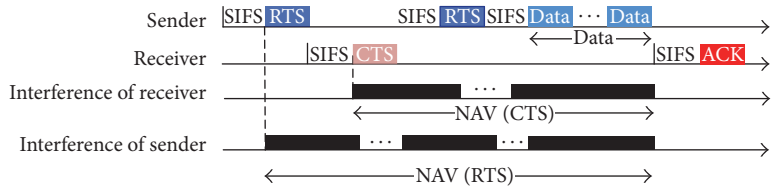



Figure 9: Broadcast protocol process.

frame (14 B per frame) intervals. Thus, the control frame cost of unicast $c_1 = 48$. When the set of packets is 2, the communication needs two RTSEP frames (58 B per frame), two CTSEP frames (15 B per frame), and two ACK frames (14 B per frame). So its control frame cost $c_2 = 174$, and its control frame cost benefit $g_2 = -0.81$. Next we consider the set of 4. The communication demands two RTSEP frames (102 B per frame), two CTSEP frames (15 B per frame), and two ACK frames (14 B per frame). In this case, the control frame cost $c_4 = 320$, and its control frame cost benefit is $g_4 = -0.67$. As we can see, the control frame cost benefit is negative in our Medium Access Control Protocol. This result means that we cost more communication resource in control frame.

However, in the above simple example, we only consider the control frame cost and ignore the total length of data frame. Note that the unicast and broadcast flows are totally different. Unicast flows cannot be encoded together with any other flow. Then, let $k$ be the demand of the flows length and let $c'$ be the cost of the set of packets. And we have

$$c'_n = c_n + k. \tag{9}$$

Similarly, let $g'$ be the cost benefit of the set of packets, and we have

$$g'_n = \frac{nc'_1 - c'_n}{nc'_1}. \tag{10}$$

Take $n = 4$ as an example; (10) turns to $k$:

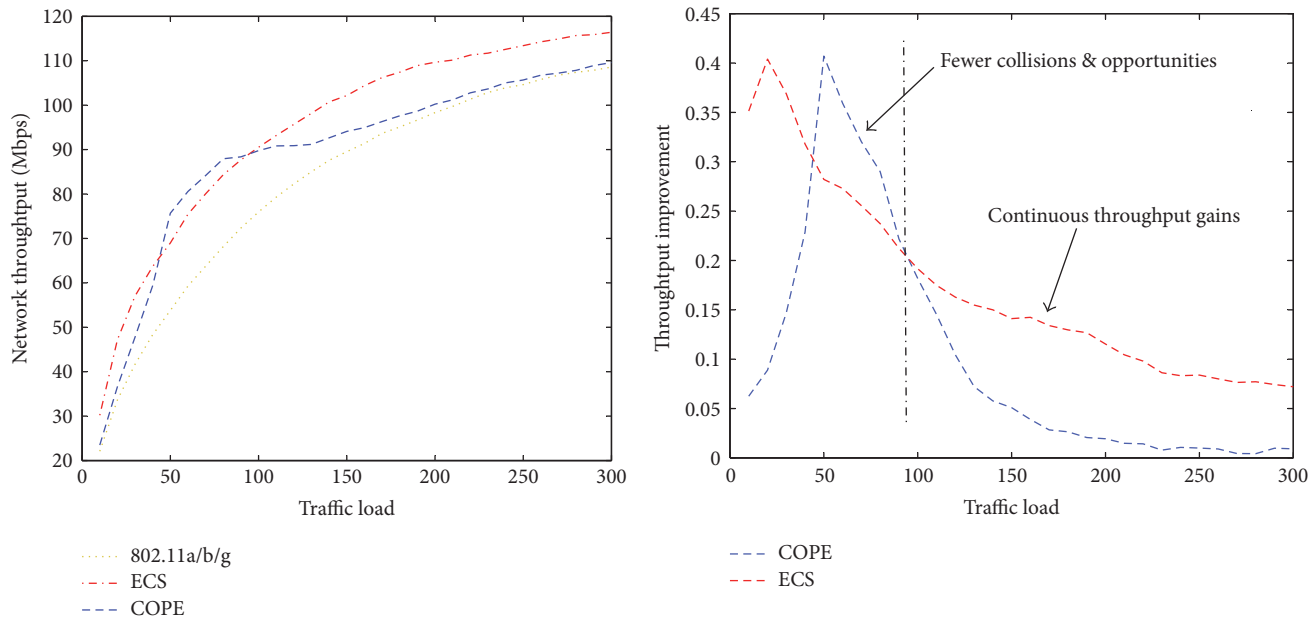
$$g'_4 = \frac{3k - 128}{4k + 192}. \tag{11}$$

When $k > 43$, the cost of the amount of information is positive in a coding scheme, approximating 0.75 finally.

## 7. Simulation and Result Analysis

Our simulation goes as follows. Flow packets are supposed to arrive according to a Poisson process. $\lambda = 100$ (unit: packets/per second). Senders and receivers are selected randomly using the uniform distribution law. We randomly distribute 86 nodes in a 1400 m × 600 m space, and the communication radius of nodes is 175 m. CSMA/CA protocol is used in MAC layer and minimum-hop routing protocol is used in network layer. The channel capacity is 54 Mbps. And the topology is set up randomly. The expectations of packet length are set to 1000 B, and each node generates $\omega$ packets every second. We control network load through $\omega$. That is to say, the rate is fixed to rate = $86\omega/8$ bps. We repeat every group of simulation for 1000 times, lasting for 2 hours, and we use the expectations as the final results. We compare our scheme, ECS, with widely used 802.11a/b/g (i.e., no network coding) and scheme like COPE with greedy algorithm in throughput augment and buffer occupancy.

Note that all the throughput mentioned in this paper is information throughput, a brand new metric to measure the transmission capability of networks. Usually, we employ network throughput to do the job, which is defined as the amount of packets that the total nodes received correctly in a particular period of time. But this classical metric fails to reflect the advantages of network coding schemes (especially COPE). One of the reasons is that nodes regard the packets that cannot be decoded as incorrect packets, but most of these packets have to be delivered correctly. Our metric, information throughput, is defined as the amount of information content that the total nodes sent in a particular period of time. It is apparent that the two metrics are exactly the




(a) Network throughput

(b) Throughput improvement

Figure 10: Throughput and its improvement comparison with different network loads.

same when nodes communicate with each other only by unicast, regardless of delivery probability. Furthermore, the new metric can represent the tradeoff between bandwidth and delivery probability.

*7.1. Network Load.* The network load is driven by $\omega$ ($\omega \in \{10, 20, 30, 40, 50, 100, 150, 200, 250, 300\}$) with 802.11a/b/g, COPE scheme, and ECS scheme. As load increases, information throughput of three schemes increases in general. When $\omega \in (50, 100)$, the interference is scarce; thus CSMA/CA protocol can provide COPE with high-quality service. Consequently, the information throughput in this period is slightly higher than ECS, and 802.11a/b/g provides lowest throughput. As load increases, ECS maintains higher throughput gains than COPE, whose throughput gains level off, reflecting the fact that COPE cannot handle this high-level interference and collision with CSMA/CA. When $\omega >$ 200, interference surges, and since too many packets are encoded together, the performance of COPE deteriorates, approximating 802.11a/b/g. Figure 10 shows the throughput gains between ECS and COPE. Both ECS and COPE do have a suitable load to get maximum throughput gains. However, we try to design a scheme with correspondingly stable throughput gains. The above results reveal that ECS provides more than 7% improvements to 802.11a/b/g, the peak of which is 40%.

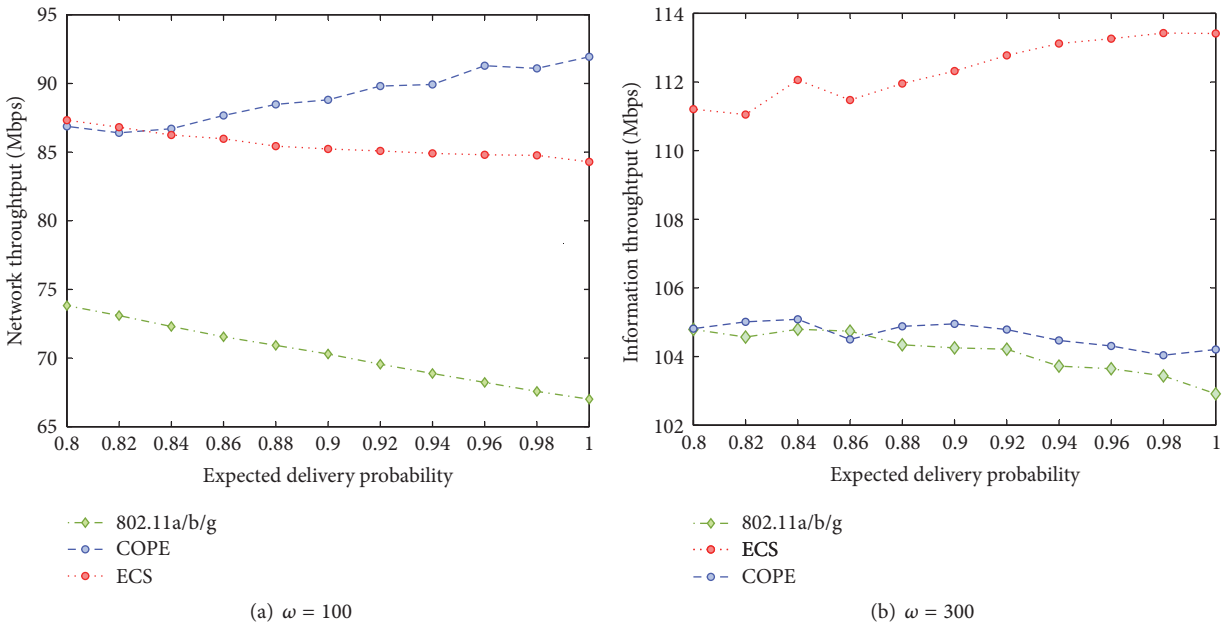
*7.2. Link Quality.* We repeat the simulation with link quality from 1 to 0.8 with the Gaussian distribution and $\omega = 100, 300$ (unit: packets per second). The flows again arrive according to a Poisson process under the assumption that network load has a slight impact on link quality.

Figure 11(a) shows that both ECS and COPE greatly improve information throughput with low load. Interestingly, with expected delivery probability decreasing, the information throughput decreases, varying from ECS and 802.11a/b/g. Note that the incorrect guess increases as expected delivery probability goes down. Receivers send request for retransmission as they cannot decode. Receivers, however, back off because they may regard this request as an unexpected interference.

Figure 11(b) plots the information throughput in high load, which is greatly shifted. ECS offers a greater improvement than COPE and 802.11a/b/g. Surprisingly, COPE has few improvements as it is close to 802.11a/b/g. Recall from Section 7.1 that the information throughput of COPE becomes flat due to the incorrect neighbor state guesses and encoding number. As link quality improves, the information throughput decreases, showing that the network has achieved its maximum capacity.

The above results reveal that COPE could not bring whole network information throughput in heavy load. It is because COPE substantially increases the utilization of nodes but decreases the sending coverage at the same time. ECS, on the contrary, makes a tradeoff between sending coverage and the utilization of nodes, which therefore yields higher throughput. Meanwhile, the model of learning neighbor state in ECS avoids incorrect decisions; thus retransmissions and long-term backoff greatly decrease. It should be noted that throughput in ECS also has a slight decrease as interference is sure to rise when multicasting encoding packets.

*7.3. Buffer Occupancy.* Many previous coding schemes assume that nodes' buffer is infinite; that is to say, infinite packets can be buffered. In practice, however, it is vital to




(a) $\omega = 100$ (b) $\omega = 300$

FIGURE 11: Throughput comparison with different link quality.

consider that buffer occupancy is limited. Compared with 802.11a/b/g, nodes maintain not only a forwarding queue but also a decoding resource queue. Data packets in the decoding resource queue may be used in decoding; otherwise, they would be discarded. As mentioned earlier in Section 2, there are two kinds of decoding resources: one is the data packets sent by nodes themselves and another one is the data packets overheard by opportunistic listening. The proportion of these two kinds of data packets is associated with node deployment location and topology. That is to say, for the nodes close to the edge of network, the queue of overhearing packets is small, and for the nodes on the trunk road of network, the vast majority of decoding resources are overheard packets. Figure 12 describes the buffer utilization of all nodes in COPE in the simulation, the horizontal axis is the node numbers, and the vertical axis is the number of data packets buffered by node when network is stable; broken line represents the number of data packets in the forwarding queue and the column represents the total amount of packets in the buffer. It can be seen from the figure that there is big difference between nodes; part of the nodes almost have no buffer decoding resources, and part of the nodes have more than 90% of the buffer to store decoding resources.

*7.3.1. Buffer Occupancy and Jitter.* We repeat the simulation with the load of 100 and 300, respectively, to summarize some feature values of buffer occupancy, including start values, end values, maximum values, and minimum values, as well as average values. We randomly pick 12 nodes and draw the K chart as shown in Figure 13. Buffer occupancy on ECS is much lower than COPE, since ECS releases invalid decoding resource. For higher load, the buffer occupancy of COPE doubles and even triples compared with medium load. Comparatively, ECS does not require much extra buffer. Also, the curve of ECS is much smoother than COPE, reflecting that ECS only jitters slightly and therefore is more stable and robust.

*7.3.2. Cost of Throughput Gains.* We repeat the simulation with load in low, medium, and heavy weight ($\omega \in \{60, 100, 200, 300\}$). The relationship between buffer occupancy and throughput gains is presented in Figure 14. Network throughput can be improved by wider frequency band with MIMO, frequency gains with advanced hardware, or our solution, network coding, which all increase cost. Distinguished from other approaches, network coding needs more computational time and storage. As we have shown in Section 5.3, ECS yields much lower computational complexity than COPE and separates the algorithm into two parts, resulting in running apart in spare time of nodes.

Almost every node exploiting ECS in Figure 14 lies in the left side, mirroring that with same throughput gains; ECS occupies fewer buffers. In addition, nodes of ECS gather together with rare variations, while nodes of COPE are more separated, indicating that ECS is more adaptive and robust. Surprisingly, some nodes (both in ECS and COPE) are distributed below zero. That is to say, not all the nodes can obtain throughput gains.

*7.4. Deployment Density.* It is significantly important to consider deployment density in wireless mesh network. For example, with intensive deployment, network is robust, but it also brings high redundancy and interference. Deploying loosely, network is in poor robustness, while the utilization of nodes is high. Thus, we evaluate the effect of deployment density on network coding scheme.

We evenly deploy $n$ nodes in every $rr$ area of one node randomly; therefore network coverage area is $r\sqrt{n} \cdot r\sqrt{n}$. We

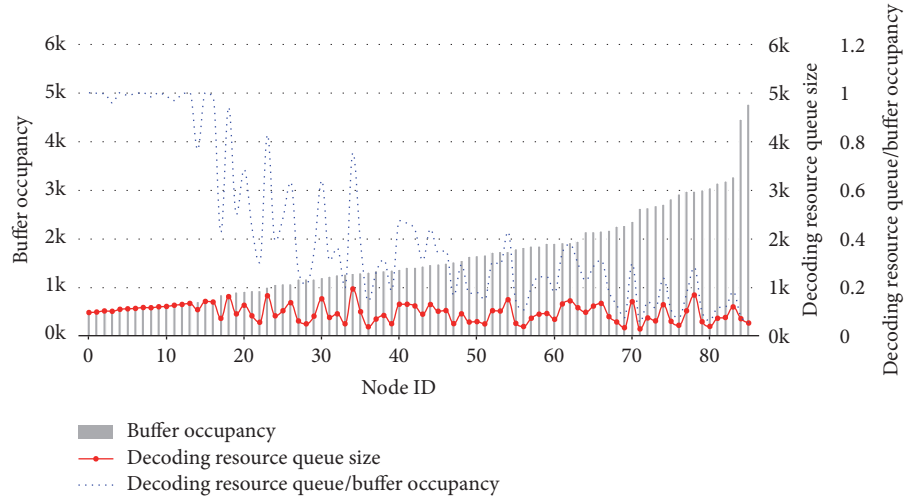
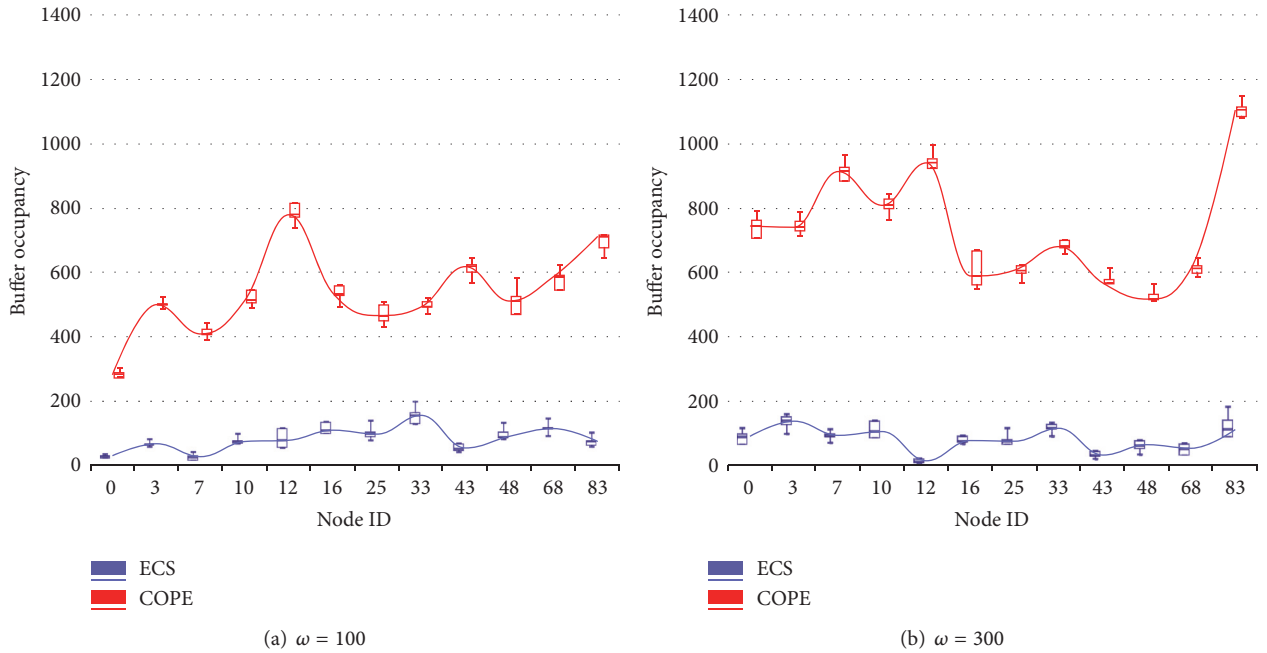



Figure 12: Buffer occupancy in COPE.




(a) $\omega = 100$ (b) $\omega = 300$

Figure 13: Buffer consumption and buffer jitter.

control deployment density by adjusting the parameter $r$. In order to ensure that the network is a connected graph, $r$ should be less than $\sqrt{2}R/2$ ($R$ is node communication radius). Meanwhile, in order to ensure that the network is totally disconnected graph, $r$ should be greater than $\sqrt{2/n}R/2$. In the simulation, we set $n = 400$, $R = 175$, and $r \in [50, 125]$. Figures 15(a)–15(c), respectively, show the network topology when the particle size is 50, 90, and 120.

Under 16 kinds of deployment density, we use coloring algorithm mentioned in Section 5, respectively, on 802.11a/b/g, COPE, and ECS. Figures 16(a)–16(i) are the results of coloring simulation.

As we can see from Figure 16, no matter whether the deployment density is intensive or not, sending coverage rate of both ECS and 802.11a/b/g is very high, while COPE is relatively low. Leveraging network coding can bring high ratio of utilization of nodes, which is mainly because every sending node conducts multicast communication with many neighbors, just as shown in Figure 16(d).

Overall, the main approach in COPE for throughput increment is to improve node utilization, which, however, consumes too much sending coverage rate. The sending coverage ratio dries up when nodes are deployed intensively in COPE. ECS, by contrast, still keeps high sending coverage

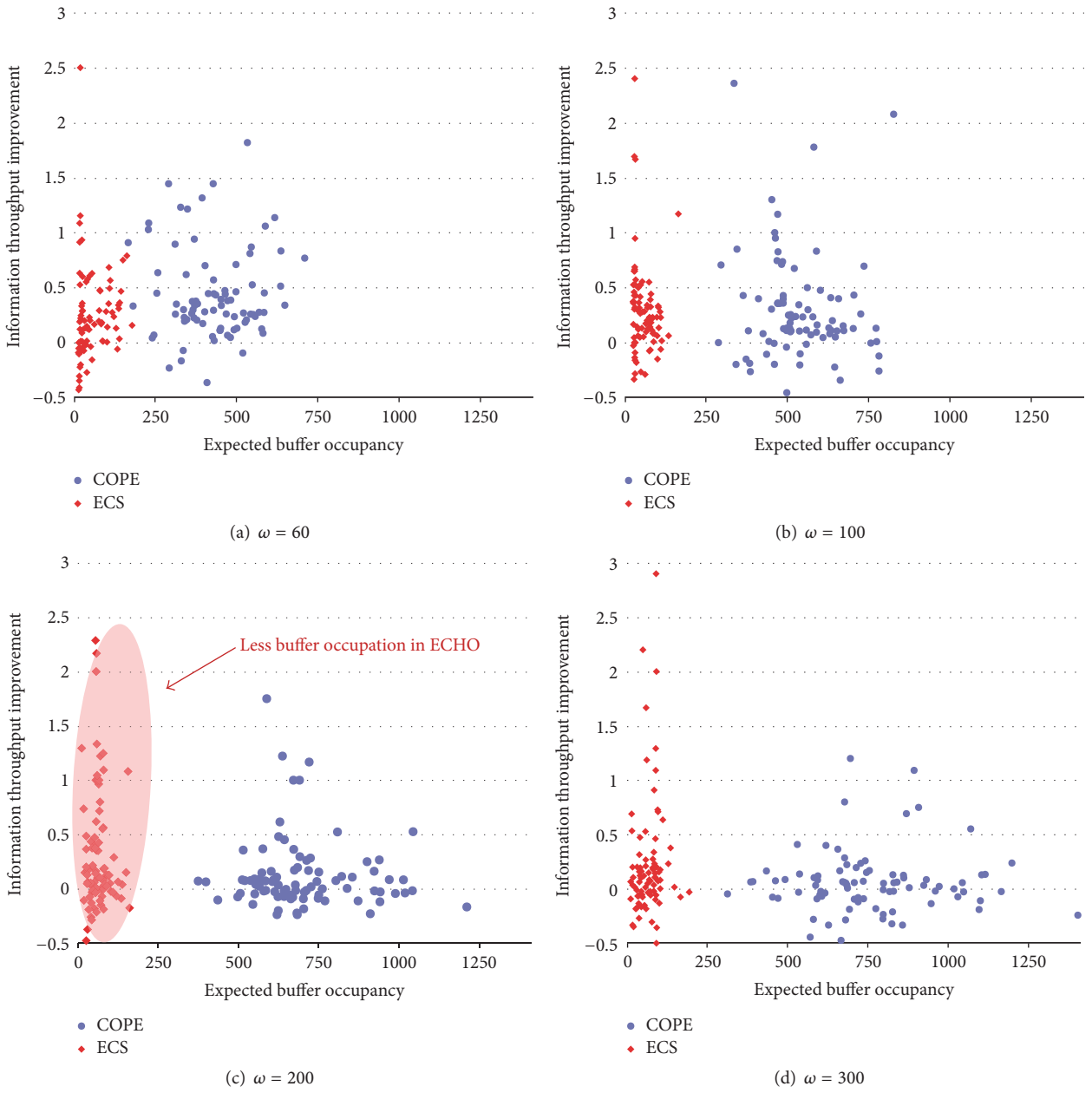


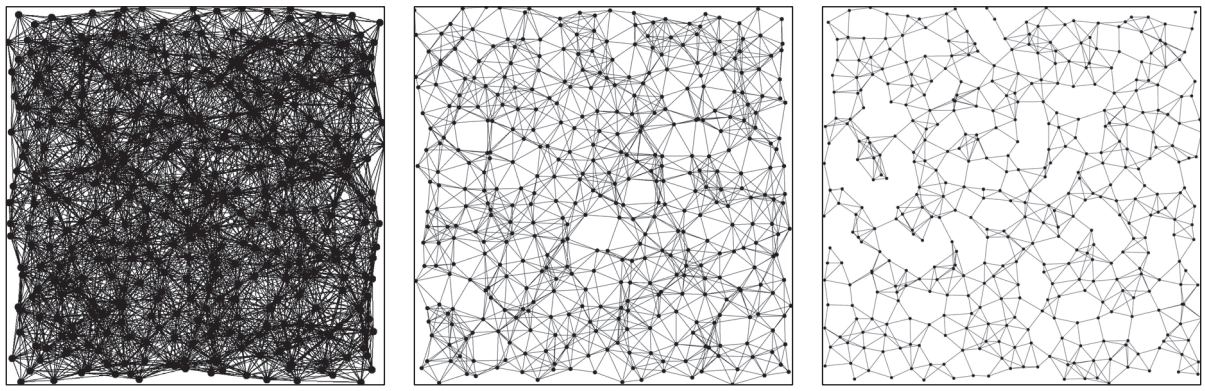

Figure 14: The overhead of throughput improvement.

ratio when nodes are deployed in high density, while the node utilization is almost the same as COPE. When the deployment density is very sparse, COPE and ECS coding are close to 802.11, because nodes barely have enough neighbors at that time; thus the coding opportunities are scarce. Figure 17 plots sending coverage ratio and node utilization under different deployment density.

No matter what deployment density is, it is apparent that ECS has similar sending coverage to 802.11a/b/g in Figures 17(a) and 17(b). At the same time, the node utilization in ECS is close to COPE. Although COPE has higher node utilization, the sending coverage ratio is much lower than ECS. Above all, there are more opportunities for network coding when the deployment is intensive; however, the interference can be very severe. But the advantage of network coding can barely take effect if the deployment is too scarce. Thus, it would be of great value in future study to find a tradeoff of encoding opportunities and deployment density.

## 8. Related Work

Due to the potential throughput benefit of network coding, it arose much attention when it was first proposed in [3]. Practical network coding schemes are aiming to increase throughput; what is vital is to design encoding algorithms and make decisions. For COPE [10], the first paper that puts



(a) $r = 50$ (b) $r = 90$ (c) $r = 120$

Figure 15: Topology.

network coding into practice has received much attention. Many follow-up researches [4, 6, 11, 17–19] are therefore addressing novel schemes to improve performance of COPE; some examples include the robust coding technique that covers the cases where COPE is oblivious of [19] and the novel scheme that allows relays forward coded packets [11]. However, the majority of them use greedy algorithm, which only focuses on the maximum packets in an encoding procedure. In fact, exploiting all opportunities of broadcasting encoded packets with greedy algorithm in a single transmission may reduce network throughput [6]. Encoding number [4] and optimality scheduling [6] aim to solve these problems, which are also the most closely related schemes to ours. However, these schemes apply for only given topologies, which is unlikely to happen in practice. In fact, it is noticeable that topologies change in wireless mesh networks is usual. Our scheme, calculating the maximum encoding number with the knowledge of topology and enlarging the sending coverage, is therefore more adaptive in mesh networks with large swaths of nodes. In particular, although [4] is also related to the upper bound of network coding, it only gives upper bound of COPE in some given topologies instead of coming up with a new approach to overcome the drawbacks of COPE. In [6], the schedule would be quite complicated when the network scales up. Every node in XOR-Sym allocates every session with a unique queue to store packets needed to be transmitted, which is very costly to maintain. Although it can achieve optimal results, it is quite complicated for scheduling algorithm to decide which sessions are allowed to transmit at a certain time. This would be aggravated when more nodes join the network. Moreover, topologies are more likely to be dynamic and arbitrary in wireless network. Therefore, fixed routing (the assumption of XOR-Sym) may not be guaranteed.

Another body of work has been looking for the improvements of network coding and decoding part based on protocols [8, 20, 21], all of which leverage opportunistic listening. Among these approaches, there are two main approaches to obtain decoding resource knowledge, namely, what native packets the neighbors should buffer to decode encoding packets: (i) transmit control information alone and (ii) via opportunistic listening. The first approach is simple and reliable. However, sacrificing throughput to transmit control information is costly, since network coding aims to augment throughput. Thus, many researches resort to the second approach, namely, exploiting ETX metric [12], to learn neighbor state, which is corrected by ROC check [22]. Typically, [18] works in lossy wireless networks with error-correcting capabilities. However, we prove in Appendix that it remains an inherent error rate, which leads to an incorrect guess and retransmission, therefore reducing throughput increment. Therefore, we elaborate on our approach, carrying coding/decoding information in RTS and CTS, which reduces communication cost, increases reliability, and consequently ensures the throughput gains.

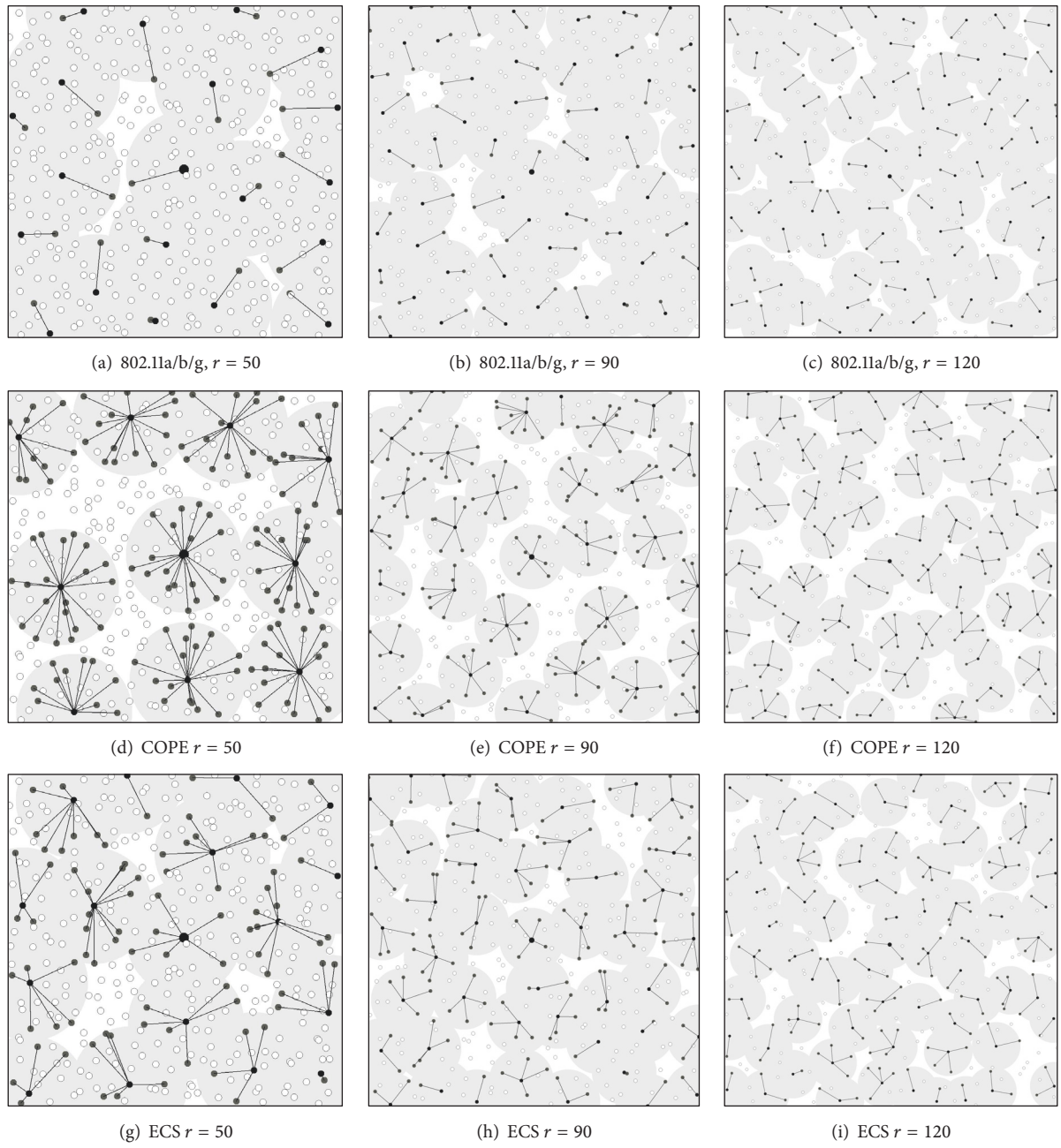
## 9. Discussion

In this section, we focus on designing an efficient and simple buffer management scheme, so that the similar throughput can be obtained with less buffer occupancy.

In COPE, opportunistically listened packets are stored in the buffer for a fixed time $T$ (default 0.5 s). After a timeout, the packet would be removed. However, such a design is not applicable, since the buffer occupancy is affected by both network load and deployment topologies. The heavier load is, the more packets of neighbors can be opportunistically listened to. Therefore, if the scheme lacks buffer monitoring, the buffer in nodes may surge, which results in no space for forwarding packets. This will be a huge disaster for the network.

The buffer occupancy in summary can be divided into two categories just as we have mentioned in Section 2. We mainly aim to manage the overheard buffer, which is in large amounts in wireless mesh networks.

Our solution, simply speaking, is to efficiently manage decoding resource by releasing the invalid buffer as soon as possible. To better illuminate our idea, let us recall the figure in Figure 2; Alice sends packet $p_1$ to relay successfully and then moves packet $p_1$ from forwarding queue to decoding resources queue. At the same time, John and Sally both



(a) 802.11a/b/g, $r = 50$ (b) 802.11a/b/g, $r = 90$ (c) 802.11a/b/g, $r = 120$

(d) COPE $r = 50$ (e) COPE $r = 90$ (f) COPE $r = 120$

(g) ECS $r = 50$ (h) ECS $r = 90$ (i) ECS $r = 120$

Figure 16: The result of deployment density coloring.

receive packet $p_1$ by overhearing the transmission and store it in the respective decoding resources queue. In COPE, all of three persons remove packet $p_1$ from the resource queue in 0.5 seconds.

In this paper, relay searches for packet $p_1$'s inverse packet in forwarding queue immediately after receiving packet $p_1$. If it is found, relay encodes packet $p_1$ firstly. Whether or not packet $p_1$ is encoded, relay will send ACK frame to Alice. If packet $p_1$ is encoded, relay sends ACKEP control frame; otherwise, it sends original ACK frame. Just as Section 6 demonstrates, ID1 stores packet $p_1$'s identification and ID2 stores identification of packet $p_1$'s inverse packet.

If all of them receive or overhear the ACKEP frame sent by relay, they know that packet $p_1$ has been coded and needs to be preserved. In this situation, packet $p_1$ is removed only in the following conditions: (i) node decodes successfully using packet $p_1$ and (ii) receiver gets data packet in ID2 field of ACKEP frame. What if packet $p_1$ cannot be encoded? For example, one of them announces to relay that he/she has no chance to overhear packet $p_1$; therefore relay can only unicast packet $p_1$ and inverse packet of packet $p_1$.

If Alice, Sally, and John monitor that relay has sent an unmodified ACK frame, they would be sure that packet $p_1$ is not encoded. In this case, they keep packet $p_1$ in a $\theta$ period

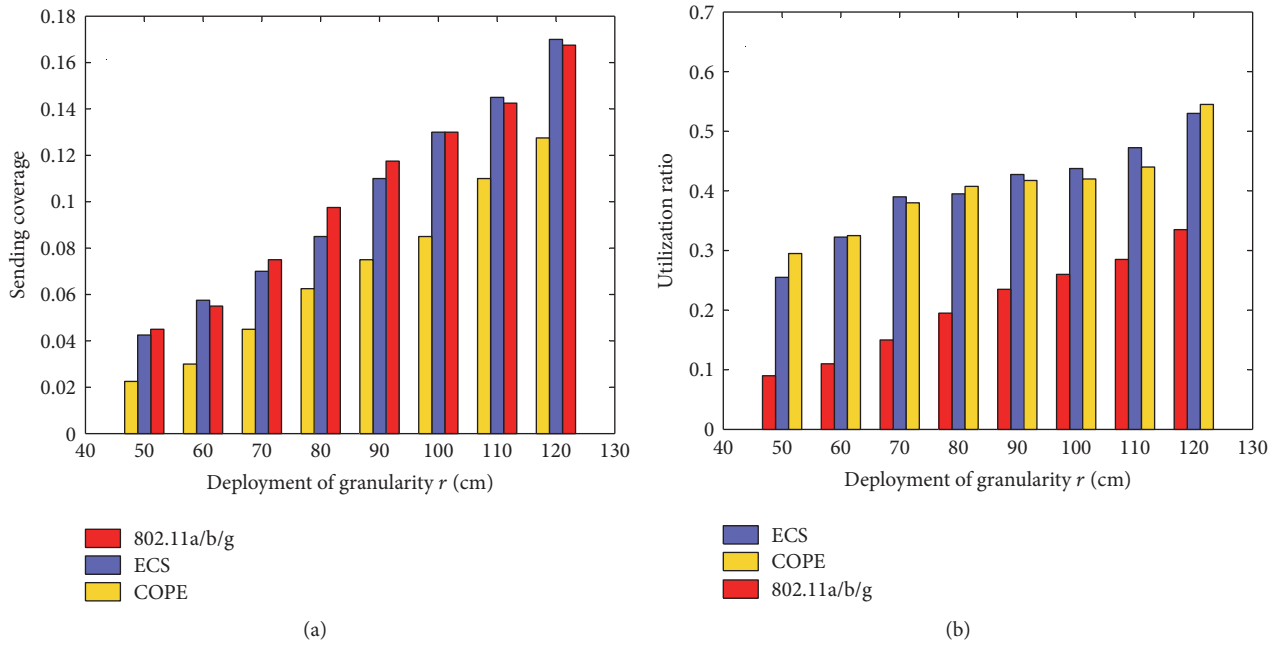



Figure 17: Deployment density with sending coverage.

of time. If they receive other ACKEP frames, whose frame of ID2 domain is packet $p_1$ in a $\theta$ period, it is shown that packet $p_1$ has been encoded with the packet whose ID number is ID1. Thus, receivers continue to keep packet $p_1$ until they decode the native packet $p_1$ or they receive packet $p_1$'s inverse packet. If these nodes do not receive ACKEP that contains packet $p_1$'s ID, packet $p_1$ would be directly removed from resource, so that packet $p_1$ is no longer encoded.

## 10. Conclusion

In this paper, we provide an efficient and low-occupancy interference-awareness model to decrease interference and make full use of nodes in mesh wireless networks. We also prove that the original ETX metric used in opportunistic listening has an inherent error ratio that would lead to decoding failure. Our solution of carrying decoding and encoding information in RTS and CTS is a low-cost way. Our model and findings can be useful for future wireless mesh networks study.

## Appendix

## Proof of Inherent Error

Let us consider the X-topology. Alice wants to send her packets to Bob and needs the relay to forward the packets. When Alice is sending packets, John overhears the packets. *link—ETX* is the expected number of transmissions. And the delivery ratio means that the transmission is successfully received and acknowledged. The relay nodes can measure the delivery ratio for each link, which is the reciprocal of $link—ETX_i$ in the communication. Assume that $link—ETX_{(aj)}$ from Alice to John is two; thus John has a 50% delivery ratio to overhear packet $p_1$ successfully. Relays consider whether packet $p_1$ and packet $p_2$ are coded together to improve throughout, calculating with $link—ETX_{(aj)}$ and $link—ETX_{(bs)}$. Let $r_1$ be the delivery ratio of packets from Alice to John and let $r_2$ be the delivery ratio of packets from Bob to Sally. Assume that there is no relation in the delivery for each link. Then, in COPE, John follows (A.1) to make decision:

$$r_1 r_2 \geq G, \tag{A.1}$$

where constant $G$ is a threshold and the default value is 0.8 in COPE.

It works well when the links are in high quality. However, it remains a primary challenge to work in lossy wireless networks. Based on the ROC curve, $r_1$ is the delivery ratio of John overhearing packets that relay considers. $1 - r_1$ is the unsuccessful ratio. Then the accuracy of this forecasting is

$$r_1^2 + (1 - r_1)^2 = 2r_1^2 - 2r_1 + 1. \tag{A.2}$$

As we can see in Figure 18, the curve is an upward parabola. When the link is in very high quality or very poor quality, the forecast is highly reliable. But the accuracy in the middle is poor. When $r_1$ equals 0.5, which reaches its valley, the accuracy is only 0.5. That is to say, if the probability of overhearing packets of John is 0.5, the relay can make a correct guess in John's ratio by half. This is just the delivery ratio of only one packets forecast. To make a coding decision, we need at least two forecasts. In another scenario with wheel topology, we need to overhear for 8 times.

Let $S$ denote the set of opportunistic listening which needs to be forecast, and let $m$ be the size of it. Each $s_i \in S$ has one Alice and one listener Bob. Next, let $p_i$ denote the delivery

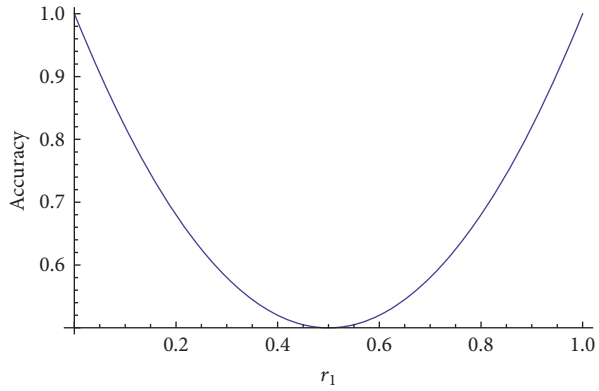



Figure 18: Accuracy of opportunistic listening in COPE.

ratio of link from Alice to Bob. In those network coding schemes such as COPE, if

$$\prod_{i=1}^{m} p_i \geq G, \tag{A.3}$$

the packets contained in $S$ are encoded together, and the incorrect ratio is

$$1 - \frac{\prod_{i=1}^{m} p_i^2 + \sum_{i=1}^{m} \left[(1-p_i)^2 \prod_{j=1}^{i-1} p_j^2\right]}{\prod_{i=1}^{m} p_i}. \tag{A.4}$$

*Proof.* Consider transmit node forecast opportunistic listening for $m$ times in turn. Let $\overleftarrow{p_i}$ be the correct ratio of forecast from $s_1$ to $s_i$. Similarly, $\overrightarrow{p_i}$ defines the correct ratio of forecast from $s_{i+1}$ to $s_m$. Then, let $p$ denote the correct ratio of all the forecast in $S$. There are four cases when the decision model examines each packet (see Table 2)

And we have

$$\overleftarrow{p_1} = {p_1}^2 + (1-p_1)^2. \tag{A.5}$$

However, if the first opportunistic listening does not meet the condition, it cannot forecast the following listening correctly. Thus,

$$p = (1-p_1)^2 + {p_1}^2 \overrightarrow{p_1}. \tag{A.6}$$

We define the result as fault if the forecast in $S$ fails more than one time. Hence,

$$\begin{aligned} p &= (1-p_1)^2 + {p_1}^2 \left[(1-p_2)^2 + {p_2}^2 \overrightarrow{p_2}\right], \\ p &= (1-p_1)^2 + {p_1}^2 \left(1 - {p_2}^2\right) \\ &\quad + {p_1}^2 {p_2}^2 \left[(1-p_3)^2 + {p_3}^2 \overrightarrow{p_3}\right]. \end{aligned} \tag{A.7}$$

Obviously, $p = \overleftarrow{p_m}$ and $\overrightarrow{p_m} = 1$; thus

$$p = \prod_{i=1}^{m} p_i^2 + \sum_{i=1}^{m} \left[(1-p_i)^2 \prod_{j=1}^{i-1} p_j^2\right]. \tag{A.8}$$

Table 2

| Probability | Reality | Prediction | Decision |
|---|---|---|---|
| ${p_i}^2$ | Success | Success | Correct |
| $(1-p_i)^2$ | Failure | Failure | Correct |
| $p_i(1-p_i)$ | Failure | Success | Incorrect |
| $(1-p_i)p_i$ | Success | Failure | Incorrect |

Therefore, the constant error rate in COPE learning neighbor state is

$$1 - \frac{\prod_{i=1}^{m} p_i^2 + \sum_{i=1}^{m} \left[(1-p_i)^2 \prod_{j=1}^{i-1} p_j^2\right]}{\prod_{i=1}^{m} p_i}. \tag{A.9}$$

□

## Disclosure

This work was first published in *ACM MSCC15, 2015,* and the differences between this extended version and the published one are fully explained in Supplementary Material available online at https://doi.org/10.1155/2017/4974165.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was based on Projects 61572402, 61672428, 61672427, and 61170218 supported by NSFC and 2012JQ8049 supported by Natural Science Basic Research Plan in Shaanxi Province of China. They also want to thank three authors (Dr. Chen Liu, Xiaoyan Yin, and Tianzhang Xing) for their contributions in the conference version of this paper. Chen Liu helped them design the algorithm in the MAC layer, which is removed from this version. Xiaoyan Yin and Tianzhang Xing ran the simulations in this part. Although they are excluded from this version because of the removal of the algorithm, they still make great contribution to this paper.

## References

[1] A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion, "Ix: A protected dataplane operating system for high throughput and low latency," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)*, pp. 49–65, Broomfield, Colo, USA, 2014.

[2] S. Sen, W. Lloyd, and M. J. Freedman, "Prophecy: using history for high-throughput fault tolerance," in *Proceedings of the 7th USENIX Conference on Networked Systems Design And Implementation (NSDI '10)*, pp. 345–360, San Jose, Calif, USA, 2010.

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

[4] J. Le, J. C. S. Lui, and D. M. Chiu, "How many packets can we encode?—An analysis of practical wireless network coding," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (INFOCOM '08)*, pp. 1040–1048, Phoenix, Ariz, USA, April 2008.

[5] J. Liu, D. Goeckelt, and D. Towsley, "Bounds on the gain of network coding and broadcasting in wireless networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (IEEE INFOCOM '07)*, pp. 724–732, Barcelona, Spain, May 2007.

[6] P. Chaporkar and A. Proutiere, "Adaptive network coding and scheduling for maximizing throughput in wireless networks," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '07)*, pp. 135–146, Montreal, Canada, September 2007.

[7] T. Ho, M. Medard, R. Koetter et al., "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[8] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-level network coding for wireless mesh networks," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*, vol. 38, pp. 401–412, Seattle, Wash, USA, August 2008.

[9] P. Li, S. Guo, S. Yu, and A. V. Vasilakos, "CodePipe: an opportunistic feeding and routing protocol for reliable multicast with pipelined network coding," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 100–108, Orlando, Fla, USA, March 2012.

[10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 36, pp. 243–254, Pisa, Italy, September 2006.

[11] S. Omiwade, R. Zheng, and C. Hua, "Butterflies in the mesh: lightweight localized wireless network coding," in *Proceedings of the 4th Workshop on Network Coding, Theory, and Applications (NetCod '08)*, pp. 1–6, Hong Kong, China, January 2008.

[12] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[13] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proceedings of the 26th IEEE International Conference on Computer Communications (IEEE INFOCOM '07)*, pp. 1163–1171, Barcelona, Spain, May 2007.

[14] Y. Lin, B. Li, and B. Liang, "CodeOR: opportunistic routing in wireless mesh networks with segmented network coding," in *Proceedings of the 16th IEEE International Conference on Network Protocols, (ICNP '08)*, pp. 13–22, Orlando, Fla, USA, October 2008.

[15] A. Khreishah, C.-C. Wang, and N. B. Shroff, "Cross-layer optimization for wireless multihop networks with pairwise intersession network coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 606–621, 2009.

[16] A. Khreishah, I. M. Khalil, and J. Wu, "Distributed network coding-based opportunistic routing for multicast," in *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '12)*, pp. 115–124, South Carolina, SC, USA, June 2012.

[17] F. Zhao and M. Médard, "On analyzing and improving COPE performance," in *Proceedings of the Information Theory and Applications Workshop (ITA '10)*, pp. 317–322, San Diego, Calif, USA, February 2010.

[18] H. Seferoglu, A. Markopoulou, and K. K. Ramakrishnan, "$I^2$NC: intra- and inter-session network coding for unicast flows in wireless networks," in *Proceedings of the IEEE INFOCOM 2011*, pp. 1035–1043, Shanghai, China, April 2011.

[19] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, "Practical network coding in wireless networks," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '07)*, pp. 306–309, Montréal, Québec, Canada, September 2007.

[20] X. Zhang and B. Li, "Optimized multipath network coding in lossy wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 622–634, 2009.

[21] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of the ACM SIGCOMM Conference on Computer Communications*, vol. 37, pp. 169–180, Kyoto, Japan, August 2007.

[22] J. R. Beck and E. K. Shultz, "The use of relative operating characteristic (ROC) curves in test performance evaluation," *Archives of Pathology and Laboratory Medicine*, vol. 110, no. 1, pp. 13–20, 1986.