Enhanced Ripple (E-Ripple) Protocol
for Chain-based Multihop Wireless Networks

Quang Vu Nguyen* and Ray-Guang Cheng**
* Department of Computer Science and Information Engineering
** Department of Electronic Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan, R.O.C.
Email: crg@mail.ntust.edu.tw

Abstract

It has been shown that IEEE 802.11 DCF medium access control (MAC) protocol may not be suitable for multihop wireless networks due to its relatively low throughput and severely unfairness problems. A token-passing MAC protocol, named Ripple, has been proposed to enhance the throughput of DCF by utilizing spatial reuse. However, two perfect assumptions of ‘fixed-length data packet’ and ‘identical interference range and transmission range’ in Ripple make it less practical in the real world. This paper proposes an Enhanced Ripple (E-Ripple) protocol aiming to remove the two restrictions. An analytical model is presented to estimate the performance of E-Ripple and the accuracy of the analysis is then verified via computer simulations. The results indicate that E-Ripple performs well in such a non-perfect environment.

1. Introduction

A multihop wireless network is a wireless backbone network offering wireless access services in outdoor environments via fixed wireless relay nodes. Fig. 1 presents the network architecture of ‘Wifly Wireless Broadband Network Service,’ a public multihop wireless network installed along main streets in Taipei City. In this chain-based multihop wireless network, the node connected to the wired Internet is the gateway (GW), while other access points (APs) act as relay nodes. Hence, the environments are characterized by the multihop radio broadcast, and spatial reuse property can be adopted to improve the network throughput [1].

In such a multihop wireless network, a common adopted medium access control (MAC) protocol is IEEE 802.11 distributed coordination function (DCF) [2]. However, DCF was designed for single hop communication. It suffers from inefficient usage of network resource due to back-off and packet collisions in the multihop environment. The carrier sense multiple access with collision avoidance (CSMA/CA) algorithm in DCF also omits the spatial reuse properties [1]. Li, et al. [3] showed that 802.11 MAC protocol with RTS/CTS can achieve only 1/7 of maximum attainable throughput. Therefore, several MAC protocols were proposed to enhance the throughput of DCF via spatial reuse [4, 5].

An RTS/CTS-based MAC protocol, named MACA-P [4], has been proposed to enhance the throughput of 802.11 MAC. It enables spatial reuse by aligning packet transmissions in multihop wireless networks. However, MACA-P requires synchronization among nodes for aligning their transmissions. It also assumes that a node’s radio can only interfere with nodes within its transmission range. Hence, MACA-P cannot work if the interference range of the radio is larger than its transmission range.

A token-passing MAC protocol, named Ripple [5], was proposed to enhance the throughput of DCF. Ripple utilizes the information provided by the network allocation vector (NAV) and a newly defined ready-to-receive (RTR) packet to enable simultaneous packet
transmissions. It was shown in [5] that 1/3 of maximum attainable throughput can be achieved. However, two perfect assumptions of ‘fixed-length data packet’ and ‘identical interference range and transmission range’ in Ripple make it less practical in the real world.

Fig. 2 showed the impacts of the two assumptions on Ripple. The simulation parameters are given in Table 1. Two types of data packet distributions were considered. The data packet size (in bytes) is randomly selected from \{64, 128, 256, 512, 1024, 1500\} in case I and \{1024, 1500\} in case II. Two interference scenarios of \(R_i=R\) and \(R_i=2.2R\) were investigated, where \(R_i\) and \(R\) are the interference and transmission range of a node, respectively. The end-to-end throughput of Ripple with 1024 bytes packet was illustrated as a benchmark herein. It is found in Fig. 2 that the network suffers from a relatively lower throughput if it accommodates a wider range of data packet size distribution. It is also found that the performance of Ripple is severely degraded when \(R_i=2.2R\).

This paper aims to extend the applicability of Ripple to a real environment by eliminating the two perfect assumptions. In such a real environment, packet collisions could be resulted from the asynchronous transmission of variable-length packets and/or unaware interference generated from two hops away. Hence, the main challenge of this paper is to avoid data packet collisions while maintaining the spatial reuse property. The proposed Enhanced Ripple (E-Ripple) protocol will use carrier sensing interval to determine node’s operation to avoid collisions due to variable-length data packet. It further utilizes carrier sensing to detect busy channel and forces each node to keep silence to avoid collisions resulted from unaware interference.

The rest of the paper is organized as follows: Section 2 describes the background, operations and theory analysis of E-Ripple. The simulation results will be given in Section 3. Conclusions and future works are finally drawn in Section 4.

2. E-Ripple

The notations and parameters used in this section are first defined.

- \(n\): total number of node in the chain network;
- \(d\): the distance between two adjacent nodes;
- \(C\): link capacity of the relay radio interface;
- \(R\): the transmission distance in which nodes are able to receive or overhear the packet transmission;
- \(R_i\): the carrier sensing distance in which nodes are able to sense the signal, though correct packet reception may not be available;
- \(R_s\): the interference distance in which any new transmission may interfere with the current packet reception [6];
- \(T_{Regen}\): the idle time that the first node waits to regenerate a new RTS;
- \(T_{Retrans}\): the idle time that a node should wait before retransmitting RTS;
- \(T_{Regen}\) and \(T_{Retrans}\) are configurable parameters. The optimal value of \(T_{Retrans}\) can be calculated base on the packet size distributions and commonly used in chain.
- \(T_b\): the busy interval to be assumed the transmission of a minimum length data packet. \(T_b\) must be longer than transmission time of control packets and shorter than that of data packet with minimum size;
- \(F_b\): a flag indicating the operations of a node. The value of \(F_b\) is either 1 or 0.

When a node senses busy channel, it starts a timer until the channel become idle again to meter the length of the busy interval. If the interval is longer than \(T_b\), the node would set \(F_b\) to 1, otherwise, \(F_b\) is unchanged. \(F_b\) is turned to 0 on error occurrence, reception of CTS, RTR or data packet, but on node initialization \(F_b\) is set to 1.

2.1. System model and Background

This work considers a multihop wireless network deployed in urban areas with chain topology as shown in Fig. 1, in which nodes are spaced equally in distance of \(d\) equal to transmission range \(R\). The interference range is assumed to be equal to 2.2\(R\) (i.e. \(R_i=2.2R\)). Therefore, the transmission of a node will interfere the reception of its 2-hop neighboring nodes. Simultaneous packet transmissions can only be occurred for nodes which are 3-hop from each other, which gives the minimum spatial reuse distance of four [7].

Without loss of generality, this paper only focuses on unidirectional downlink transmission, which data traffic come from the gateway to the relay nodes.
Bidirectional can be implemented by either using one radio interface with time-division duplex or two radio interfaces with frequency-division duplex [5]. Having two interfaces will also prevent node from channel assignment problem.

Ripple is a token passing MAC protocol. The tokens used in Ripple are RTS and RTR, only node holding a token can transmit RTS or RTR to initiate data exchanging. In Ripple, a node generates and circulates tokens base on its state [5].

E-Ripple is developed based on Ripple. However, modifications were made to deal with the collisions resulted from variable length packets and the increased interference range. Different to Ripple, E-Ripple uses CTS and RTR as tokens for data packet transmission. A node is allowed to transmit a data packet when it receives a token. Moreover, the generating and circulating of tokens in E-Ripple depends on the state and \( F_b \) of each node. E-Ripple further utilizes the carrier sensing functionality supported by 802.11 to detect the busy channel due to the transmission of a node that is 2-hop away. With carrier sensing, \( N_3 \) and \( N_4 \) in Fig. 1 may detect the data exchange between nodes \( N_2 \) and \( N_6 \), respectively. Therefore, a node can keep silence whenever it senses the channel as busy.

In operation, nodes sense the co-channel interference power, if the power is greater than a threshold, the channel is assumed to be busy. Hence in practice, \( R_e \) can be adjusted [6] by changing the value of power sense threshold in node’s physical layer. In E-Ripple, all nodes should adjust their sensing range \( R_i \) to be \( R_i = 2.2 R \) so that they can detect busy, interference channel and keep silent to avoid collision.

The principle of using carrier sensing technique to keep nodes off the unintentional collision is presented in Fig. 3. When node \( N_4 \) is transmitting to node \( N_5 \) at time \( t_1 \), 1-hop node (i.e. node \( N_3 \)) will overhear the transmission and keep silent in the interval of NAV, while 2-hop node (i.e. node \( N_2 \)) senses busy channel and will keep silent. Even when node \( N_1 \) gets ready and transmits RTS frame, node \( N_2 \) will not reply with token CTS; the token is lost. In order to recover the token node \( N_1 \) goes in the retransmission loop: 1) waits for \( T_{Retrans} \) and 2) transmit RTS frame. The retransmission loop will be terminated when the node gets a CTS token from its successor or detects busy channel from transmission of other nodes.

In E-Ripple, each node can be operated in any one of the five states as listed below. The state transition is triggered by a packet transmission or reception.

- **IDLE**: IDLE state is designed for initialization and fault tolerance. A node moves in this state on initialization or occurrence of error.
- **Listen 1 (L1)**: nodes in L1 state are forbidden to transmit. A node transits to this state when they finish a data packet transmission, overhear NAV from successor, or sense the channel as busy.
- **Listen 2 (L2)**: nodes in L2 state have to keep silent in NAV. A node moves in L2 state when it overhears NAV from its predecessor. The node in L2 state would transit to RX state after the expiry of NAV and the channel remains idle for SIFS.
- **Transmit (TX)**: nodes in TX state are allowed to transmit RTS to its successor to get a token for data packet. The first node (i.e. GW node) enters this state when its \( F_b \) is 1 and channel is free for \( T_{Regen} \). The other nodes enter this state after they receive a data packet.
- **Receive (RX)**: nodes in RX state are ready to receive a data packet. Therefore, a node shall transmit CTS or RTR packet to its predecessor. A node moves in RX state when it is in either L2 state and senses channel as idle for SIFS; or L1 state and receives RTS from its predecessor.

E-Ripple is a distributed MAC protocol. The operation of each node depends on its state and its flag \( F_b \). A node is allowed to transmit RTS, RTR or reply CTS only if its \( F_b \) is 1. Fig. 3 shows an example illustrating the simultaneous packet data transmissions enabled by E-Ripple. After finishing transmitting data packet, node \( N_1 \) (i.e. GW) receives ACK and also NAV from node \( N_2 \), so it will keep silent in NAV. After the NAV period, although the channel is free in the interval of time to transmit a CTS packet plus 2 short inter-frame spaces (SIFSs), \( N_1 \)'s flag \( F_b \) is 0 so it keeps silent. When node \( N_3 \) transmits data packet, \( N_1 \) detects busy channel and starts timer until \( N_3 \) finishes transmitting. Time to transmit a data packet is longer than \( T_b \), thus, \( N_1 \) turns its flag \( F_b \) to 1. After that, \( N_3 \) detects idle channel, but it will not transmit RTS until the free interval length is equal to \( T_{Regen} \). If a CTS frame from node \( N_2 \) is not received correctly, \( N_1 \) goes in the mentioned retransmission loop. It can be found in Fig. 3 that the regeneration interval should be long enough for a node to conduct a minimum-length data packet transmission with its neighbors. That is, \( T_{Regen} \geq T_{ACK} + T_{CTS} + 3SIFS + T_{data\_min} \).

Besides carrier sensing technique and retransmission, the cut-through scheme presented in [8] is also adopted in E-Ripple to reduce the RTS/CTS handshake overhead. The key point of cut-through approach is to conjoin RTS for the successor with ACK for the predecessor. This combined ACK frame adds extra 6 bytes to carry the MAC address of the successor taken from RTS; the 2 bytes duration has the value of the RTS duration field. This frame acts as both
Figure 3. State transition and timing diagram of E-Ripple

ACK to previous node and RTS to the next node. Hence as drawn in Fig. 3, at time $t_2$ when node $N_5$ transmits ACK to node $N_4$, node $N_6$ will receive the RTS. If $N_6$ is free and its $F_b$ is 1, it would reply to $N_5$ with a CTS packet. In the case that the cut-through scheme fail to respond the positive CTS, the node would turn in retransmission loop and the original RTS/CTS handshake.

2.2. Operations of E-Ripple

Fig. 3 illustrates the state transition and timing diagram for several first nodes in the downlink chain. After initialization, all nodes use 802.11 DCF to recognize the first (i.e. GW) and the last node in chain, broadcast the values of $T_{Regen}$ and $T_{Retrans}$, and then turn to IDLE state. E-Ripple begins with the RTS transmission of the first node; this will trigger E-Ripple operations and the state transition of all nodes. In Fig. 3, node $N_1$ receives token CTS from node $N_2$, thus it can transmit a data packet. When $N_1$ receives ACK, node $N_3$ receives RTS conjoined in this ACK from $N_2$ too. After $SIFS$ $N_3$ replies $N_2$ with the token CTS, and so on; tokens will be generated along with data transmission in the chain. Note that, on exchanging RTS/CTS to get the token, a node not only relays packets from the successor, but can also transmit its own packets.

When node $N_i$ finishes transmitting a data frame, its $F_b$ is 0. $N_i$ enters L1 state and keeps silent until its $F_b$ is changed to 1 and channel is free in the period of $T_{Regen}$ to transmit RTS. Particularly, $N_i$ overhears the NAV from node $N_2$, then senses busy channel from data packet transmission of node $N_3$, which set its $F_b$ to 1. When it senses idle channel in $T_{Regen}$ with $F_b$ is 1, $N_1$ will turn to TX state and transmit RTS. As mentioned before, if a positive CTS packet can not be received correctly, $N_1$ will enter the retransmission loop until it gets CTS token or detects busy channel. The RTR packet is used to replace CTS in case a node is in state L2 and can not receive the RTS or combined RTS packet correctly. E.g. in Fig. 3, when node $N_5$ is in state L2 and fails to receive the combined RTS from node $N_4$, after $SIFS$, it transmit RTR packet to invite $N_4$ for data transmission. By this scheme, E-Ripple creates the ripple phenomenon in the chain so that in average two nodes in distance of the spatial-reuse distance can have concurrent transmission without interfering each other.

The finite state machine, based on which the downlink transmission can be implemented, is drawn in Fig. 4. A node adopting E-Ripple enters state L2 on overhearing NAV from its predecessor. After the NAV if $F_b$ is 1 and channel is free in $SIFS$, the node would turn to RX state and transmit RTR. Otherwise, it gets in L1 state. In RX state, after receiving data packet and transmitting the combined ACK/RTS packet, the node will change its state to TX to transmit data to the next node. In L1 state, if $F_b$ is 1 and channel is free in $T_{Regen}$, it would get in TX state and transmit RTS. Without error, a node in TX state always turns to L1 after it finishes transmitting a data packet. Any error occurs to a node would bring it back to IDLE state with $F_b$ is 0.
2.3. End-to-end Throughput Analysis

Let $p_i$ and $P_i$ denote the probability and cumulative probability that a data packet has size $i$ in the range $[S_{\text{min}}, S_{\text{max}}]$. Denote $T_{\text{RTS}}, T_{\text{RTR}}, T_{\text{ACK}}, T_{\text{CTS}}$ as the time required to transmit RTS, RTR, ACK, and CTS packet. Let $T_i$ be the time required to transmit $i$ bytes data packet, which gives $T_i = i/C$. Then, the average overhead to transmit a data frame, $H$, is given by

$$H = T_{\text{ACK}} + T_{\text{CTS}} + 3S_{\text{IFS}}. \quad (1)$$

If the data length is fixed, the minimum achievable spatial reuse distance is 4; hence, the maximum number of parallel link is $n/4$. Let $N_k$ be the node sending the largest data packet size $S$ at time $t$. The probability that there is no packet longer than $S$ and at least one packet has size $S$ is

$$P^*_{S} = P_{S}^{\frac{n}{4}} - (P_{S} - P_{S})^{\frac{n}{4}}. \quad (2)$$

Without loss of generality $k$ is assumed to be equal or greater than 4 (i.e. $k \geq 4$). Consider $N_k$ and its preceding nodes (i.e. $N_{k-4}, N_{k-3}, N_{k-2}$) in the interval when the $S$ bytes data packet is sent. While $N_k$ is sending the $S$-byte packet, $N_{k-4}$ sends an $i$ ($i \leq S$) bytes one to node $N_{k-3}$. After receiving, $N_{k-3}$ transmits RTS but can not get token CTS immediately because $N_{k-2}$ is detecting busy channel from node $N_k$’s transmission. Thus, node $N_{k-3}$ has to wait for an interval that is at least the time to transmit the difference of the two packet size (e.g. $(S-i)/C$). And, the useful utilization is time to transmit the short packet divide by the time for longer one include the overhead of handshake procedure (i.e. $T_i/(H+T_S)$). This is also right for the consecutive succeeding and preceding communication links in the chain network, so the mean useful part of the each parallelism link is

$$U = \sum_{i=S_{\text{min}}}^{S} P_i \frac{T_i}{P_{S} T_{S} + H}. \quad (3)$$

Hence, the mean throughput in the interval to transmit the $S$-byte packet, $C_N$, is derived from Eq. (4). In case that the minimum achievable spatial reuse distance of 4 is considered, the lower bound of end-to-end throughput can be obtained from Eq. (5).

$$C_N^* = \frac{C}{n} \times \left[ U \left( \frac{n}{4} - 1 \right) + \frac{T_S}{T_S + H} \right]. \quad (4)$$

$$C_N = \sum_{S_{\text{min}}}^{S_{\text{max}}} P^*_S C_N^* \quad (5)$$

The end-to-end throughput in $E$-Ripple depends on packet size distribution and the chain length. If the data packet size is fixed (i.e. $i=S$ with probability $p_i=1$), Eq. (5) is reduced to Eq. (6). That is the best throughput can be achieved by $E$-Ripple with spatial reuse distance is 4. In this instance, the end-to-end throughput would not be affected by number of nodes in chain.

$$C_N = CT_S /4(T_S+H) \quad (6)$$

3. Simulation Results

In this work, the simulations were performed using $C$-based platform. Each sample was obtained by averaging 100 outcomes and each outcome was collected within 200 seconds. The simulation parameters used in the simulations are listed in the Table 1. The IEEE 802.11 PHY with $C=1$ Mbps was chosen as an example for the air interface of the relay links. As in [5], an error-free channel was considered; the traffic is only injected at the GW node, and the end-to-end throughput is measured at the tail relay node. The performance metrics of end-to-end throughput $C_N$ and end-to-end packet delay were investigated. In the simulation, the error of throughput results was less than $2 \times 10^{-3}$ Mbps within a 95% confidence level.

<table>
<thead>
<tr>
<th>Table 1. Simulation parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$d$</td>
</tr>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>$R, R_c$</td>
</tr>
<tr>
<td>PHY preamble</td>
</tr>
</tbody>
</table>
Time to transmit a data or control packet can be calculated using these parameters. E.g. the time to transmit a RTS packet is
\[ T_{\text{RTS}} = T_{\text{PHY\_preamble}} + T_{\text{20\_bytes}} = 192 + (20 \times 8) = 352 \mu s. \]
The regeneration time is
\[ T_{\text{Regen}} = T_{\text{ACK}} + T_{\text{CTS}} + 3\text{SIFS} + T_{\text{data\_min}}. \]

Fig. 5 demonstrates the end-to-end throughput of E-Ripple and 802.11 DCF for several fixed data packet sizes. The analysis result (6) about the best achieved result of E-Ripple in the previous section is validated. When the chain length is short (e.g. 2 to 4 nodes, E-Ripple’s results are not available), there is almost no contention and collision, thus, 802.11 DCF gain very high throughput (i.e. 0.85Mbps and 0.7Mbps for packet with 1024 and 512 bytes, respectively). When the number of nodes increases, throughput of 802.11 DCF markedly drops down 0.1Mbps. While E-Ripple keeps the stable capacity of 0.217 and 0.231Mbps for packets with 512 and 1024 bytes. The end-to-end packet delays of 1024 bytes packet in 8-node chain with different offered load are illustrated in Fig. 6. When the offered load is light, 802.11 DCF delay time is similar to that of E-Ripple. But when the offered load is raised, DCF’s delay dramatically increases; meanwhile E-Ripple’s stays the same.

Fig. 7 presents the analysis and simulation results of E-Ripple compared to 802.11 DCF and Ripple in the distribution of data packet size in different ranges and chain lengths. In this simulation, the interference range is considered as more than twice of transmission range (i.e. \( R_i = 2.2R \)); packet size is randomly selected in: case 1 \{64, 128, 256, 512, 1024, 1500\}; case 2 \{1024, 1500\}. Without mechanism to prevent nodes from collision and recover lost token, Ripple perform poorly (i.e. less than 0.07Mbps) in these situations. When the packet size range is large (e.g. [64, 1500]), the wasted time caused by the packet size differences is large, and thus lessens E-Ripple’s performance. When chain length increases, the number of simultaneous transmissions with different data sizes, hence wasted waiting time also, increase to reduce protocol’s performance. But in average, E-Ripple performs twice better than 802.11 DCF.

4. Conclusions and Future Works

This paper proposes an Enhanced Ripple (E-Ripple) MAC protocol for chain-based multihop wireless network. E-Ripple can perform well to prevent nodes from collisions, and preserve spatial reuse property of Ripple in the situations that one node can affect to 2-hop far away nodes and the packet size is variable. The analytical model for computing the end-to-end throughput with different data packet size distribution is also presented. Simulation results demonstrate the accuracy of analysis and indicate the advantages of E-Ripple in the chain-based multihop wireless networks.

Currently, we are further investigating the performance of E-Ripple using IEEE 802.11a with 54 Mbps as the air interface of the relay links. The future work of this research is to improve end-to-end throughput and generalize E-Ripple to fit the tree and ring network topology in wireless mesh networks.
Acknowledgement

This work was supported in part by the National Science Council, Taiwan, under Contracts NSC 97-2219-E-011-007 and NSC 97-2221-E-011-122.

References


