An Energy Saving MAC Protocol For Sensor Networks

Yu Zhou\textsuperscript{1*}, Fanchun Jin\textsuperscript{1}, Suresh Subramaniam\textsuperscript{2}, and Hyoong-Ah Choi\textsuperscript{1}

\textsuperscript{1}Department of Computer Science
\textsuperscript{2}Department of Electrical and Computer Engineering
The George Washington University
Washington, DC
Emails: \{yzhou,jinf,suresh,hchoi\}@gwu.edu
*Corresponding author

Abstract: As the energy consumption in sensor nodes is dominated by the radio transmission/reception circuitry, communication protocols must be designed for economy in radio communications. Sensor nodes are generally equipped with short-range radios that have various characteristics including data rate, power consumption in transmit, receive, idle and sleep modes, and time to switch from one mode to another. These parameters can have significant effects on the performance of communication protocols in low energy sensor networks. In this paper, we present a protocol using RTS-Tone which is based on two-radio STEM-Tone mechanism Matthew (2005) and standard 802.11 MAC protocol. In our protocol, using only one radio, a sender uses multiple successive RTSs to simulate a STEM tone which can wake up multiple destination nodes. All nodes periodically wake up and try to detect the RTS-Tone on the channel and periodically go to sleep to save energy. For performance analysis, we have conducted ns2-based simulations. The results show that our protocol outperforms the STEM-Tone based protocol over three performance metrics: energy consumption, packet latency and packet drop rate.

Keywords: sensor nodes, energy-saving, power consumption, RTS-tone, STEM-tone, MAC protocol

1 Introduction

Wireless sensor networks are ad hoc networks composed of a large number of small sensor nodes. A sensor node collects sensory data from its environment, and such data are processed by the node circuitry to determine if an “event” has occurred. Once an event occurs, the node needs to react correspondingly to send or receive data. Due to the small capacity of a sensor node and other physical reasons, the energy of the sensor node is limited and usually cannot be renewed. Thus, the lifetime of a network is constrained by the amount of energy that is spent by the
sensor nodes in performing their operation of sensing, processing, and transmitting and receiving data. Furthermore, the power consumption in sensor nodes is dominated by the radio transmission/reception circuitry. Hence, the communication protocols must be designed such that the energy consumed by the radio circuitry is as small as possible.

In the wireless sensor network, energy consumption, average packet latency and packet drop ratio are several key factors to evaluate the whole network performance and efficiency. But generally speaking, energy consumption and average packet latency are contradictory: less energy consumption means usually more packet latency, and vice versa. To achieve an overall good performance, we need find a good balance between them. Each node's radio can be in one of four states: sleeping, idle, transmitting and receiving, and the radio powers in each state are different. For Mica2 Mote Sensors in MICA2 (2004), the transmitting power is 81mW, the receiving and idle powers are both 30mW, the sleeping power is 0.003mW. The ideal energy saving protocol is to wake up the node when there is an in-coming or out-going packet, and meanwhile the wireless channel is free as well, and to let the node sleep at other times. This is the lower bound on energy consumed for various energy saving protocols.

Our energy saving protocol is focused on the Medium Access Control (MAC) layer and works as follows. Initially, each node periodically and alternately wakes up and goes to sleep. When a node tries to start a data transmission, it will send out one RTS-Tone first, which is composed of multiple successive RTTs. The destination nodes’ addresses are piggybacked in each RTS. The duration of the RTS-Tone is long enough such that every neighbor node can detect at least one RTS in the periodic awake state. Only the addressed destination nodes will keep awake once they detect the RTS-Tone; other neighbor nodes will go to sleep and wake up again when the whole transmission is over.

In Section 2, we present energy saving protocols available in the literature. The protocol in Matthew (2005) is discussed in details as it is closed related with ours. Section 3 describes our protocol in detail. Section 4 presents simulation results in which we apply our protocol in multiple-hop and multiple-flow topologies individually and compare the performance of our protocol with Matthew (2005). In order to show the performance difference fairly and clearly, we use the same topologies and technical parameters. Conclusions are given in Section 5.

2 Related Work

A good wakeup schema is essential to the performance of an energy saving mechanism. When to turn on the radio and when to turn off the radio is the biggest issue in the wakeup schema. STEM (Sparse Topology and Energy Management) Curt (2002) is an energy management schema which trades tolerant packet latency for energy consumption. In STEM, usually data packets and control packets are sent out through two different radios individually, data radio and wakeup radio, which use two separate frequency bands to avoid collisions. STEM-Beacon and STEM-Tone are two major mechanisms in STEM.

In STEM-Beacon mode, firstly the sender will send out a beacon to all its neighbors, then wait for the acknowledgment from the destination receiver node;
if no acknowledgement is received, the sender will repeat the whole process until
gets one or the max retransmission threshold is reached. To ensure the receiver can
catch at least one beacon during the listening period, the relation of different time
periods should satisfy $T_{\text{listen}} \geq 2T_{\text{beacon}} + T_{\text{wait}}$, and $T_{\text{wait}} > T_{\text{ack}}$ as in Fig. 1.

In STEM-Tone mode, the sender will send out a tone control message which
is long enough to notify all the neighbor nodes. The way that the tone notifies
the neighbors is different from the beacon. The beacon contains the address of the
destination node; only that node will respond using an acknowledgement message.
But a tone doesn’t contain any meaningful information about the destination; all
the neighbors can only sense the channel is busy if tone is being broadcast. Each
node sensing the tone will be woken up accordingly. In STEM-Tone, to ensure
the receiver can detect the tone, the time durations for STEM-Tone need to satisfy
$T_{\text{tone}} \geq 2T_{\text{listen}} + T_{\text{sleep}} + T_{\text{trans-off}} + T_{\text{trans-on}}$, where $T_{\text{tone}}$, $T_{\text{listen}}$, $T_{\text{sleep}}$, $T_{\text{trans-off}}$, and $T_{\text{trans-on}}$ are defined as shown in Figure 2.

A two-channel energy saving MAC protocol based on data rate estimation
(STEM-Tone-RATE-EST) is discussed in Matthew (2005). In STEM-Tone-RATE-
EST, each sensor node has two radios — data radio and wakeup radio, and each
radio has a separated channel, named as primary channel and wakeup channel indi-
vidually. The primary channel is used to send and receive data packets, and
tone packet is sent out through wakeup channel. Each node having detected the
tone on the wakeup channel will turn on its data radio. Once the tone is over,
the sender will continuously send out a filter packet, which contains the destination
addresses through the data channel to only inform those destination nodes to keep
awake; other nodes will turn off their radios and go to sleep. The later
packet exchange between the sender and the receiver follows the standard 802.11
MAC protocol, that is to say, the standard 802.11 MAC protocol is encapsulated in
protocol STEM-Tone-RATE-EST, and the data radio is always awake when packet
exchange is running under the standard 802.11 MAC protocol. In our protocol, we
just use only one radio for the whole process instead of two, and we don’t intro-
duce any other new packet type either. Our protocol has smaller and more flexible
controlling granularity in the MAC layer, and hence achieves better performance.

Some other MAC protocols are also presented for wireless sensor networks, such
(2004), a Pipeline Tone Wakeup (PTW) schema is presented which is achieves the
balance between energy saving and end-to-end delay. Two algorithms named Global
Schedule Algorithm (GSA) and Fast Path Algorithm (FPA) are developed in Yuan
(2005) to control and exploit the presence of multiple schedules to reduce energy
consumption and latency.

3 Protocol

3.1 Threshold Wakeup

In our new protocol, comparing with the protocol with the two radios and
two channels in Matthew (2005), we use only one radio and one channel without
introducing any new packet or message, but we borrow the wakeup mechanism from
STEM-Tone. Initially, each node behaves as a receiver because it has no packet
to send. Each node periodically wakes up to listen to the channel and tries to
detect a broadcast RTS-Tone from some sender. If no RTS-Tone available, then
\( T_{\text{listen}} \) time later, this node will go to sleep to save energy, and \( T_{\text{sleep}} \) time later,
it will wake up again. If some node \( A \) has packets to send out, it will convert
into a sender from a receiver. In order to decrease potential collisions as much
as possible during the whole data transmission in the future, node \( A \) will send
out an RTS-Tone first, which is composed of \( K(K \geq 1) \) successive RTS packets.
The continuous multiple RTSSs act as a tone of the STEM-Tone schema but they
contain meaningful information which is the destination’s address. Considering
the asynchrony of wakeup timestamps of all neighbor nodes, the duration of the
RTS-Tone needs to be long enough and satisfy the inequality below to ensure every
neighbor node can detect this tone and can receive at least one RTS packet in
RTS-Tone: \( K \times T_{\text{rts}} \geq T_{\text{listen}} + T_{\text{sleep}} + T_{\text{tran-off}} + T_{\text{tran-on}} \) (See Fig. 3).

We notice that the energy consumed by the sender will increase if \( K \) becomes
larger, but meanwhile, the energy consumed by the receiver will decrease because
the sleeping time becomes longer. Compared with traditional single RTS in the
standard 802.11 MAC protocol, the energy consumption caused by RTS-Tone is
more, thus it is necessary to decrease the frequency of occurrence of RTS-Tone. We
set a queue threshold \( L \) for every node. Only when the number of out-going packets
in the queue hits the threshold, can an RTS-Tone be sent out; this is the reason it
is called threshold wakeup. In each RTS packet of an RTS-Tone, the addresses of
L destination nodes will be piggybacked, thus one RTS-Tone can inform at most \( L \)
destination nodes to prepare for the subsequent data transmission at once. If the
sender wants to send multiple packets to the same receiver, it will put the receiver’s
address multiple times up to \( L \) in the RTS.

Besides the destination address, RTS also contains a NAV (Network Allocation
Vector) value which indicates the total transmission time of \( L \) packets to be reserved
on the channel by this node; other senders should not access the channel in order to
avoid a collision. Different RTSSs in the same RTS-Tone have different NAV values
according to the order in the RTS-Tone. The first RTS has the largest NAV and
the last one has the smallest. When the first packet transmission completes, the
later \( L - 1 \) packet transmissions of the threshold wakeup will not invoke RTS-Tone
and only exactly follow standard 802.11 MAC protocol. (See Fig. 4)

In order to save energy, the \( T_{\text{listen}} \) for each node is usually quite small compared
with transmission time of one RTS. If a node doesn’t detect an RTS-Tone when
it wakes up, it will turn off the radio after \( T_{\text{listen}} \) time, otherwise it will defer the
turning off until it gets one packet. If the in-coming packet is not an RTS packet,
it is just discarded. Suppose a neighbor node \( B \) receives an RTS from the sender
\( A \). It will then check the destination address of this RTS packet. If the address
is different from its own, node \( B \) realizes that the sender \( A \) is trying to start a
transmission with another node. After updating its NAV according to the NAV
value in the received RTS, node \( B \) will go to sleep immediately and wake up at
the time according to the updated NAV. If the receiver’s address is identical to \( B \)’s
own address, and if node \( B \) is idle as well, \( B \) can calculate the time at which it has
to send the CTS packet as follows:

\[
T = T_{\text{RTS}} i + T_{\text{round}} j
\]  \tag{1}
where $T_{RTS}$ is the transmission time for an RTS packet, $T_{rcv}$ is the total transmission time for a packet including RTS, CTS, DATA, ACK and other additional times such as IFS and propagation delay, $i$ is the number of RTSs remained in the RTS-Tone, $j$ indicates the order of the receiver’s address in the RTS packet.

Thus, node $B$ can go to sleep immediately and wake up $T$ later to receive the RTS. This is unlike the protocol in Matthew (2005) where each neighbor node has to be awake after receiving a tone; this means more energy could be saved in our protocol. Right after node $A$ completes $K$ RTS, $B$ also wakes up as preplanned and sends back CTS to $A$; once $A$ receives this CTS, the later transmission between the sender and the receiver follows the standard 802.11 MAC protocol. If node $A$’s transmission fails because of a collision with another transmission to node $B$, node $A$ will retransmit an RTS-Tone when the medium is free and a back-off period completes.

To avoid being idle for a long time, each node has a threshold duration $T_{thresh}$. When a receiver successfully receives a packet of any type, it will immediately start a timer with duration of $T_{thresh}$. If no more packet arrives or no more packet needs to be sent out when the thresh timer runs out, the receiver will turn off the radio and go to sleep to save energy. After a receiver sends back a CTS to the sender, it also starts a threshold timer. If no DATA packet arrives from the sender in $T_{thresh}$, the receiver will go to sleep. Both the sender and the receiver will start a threshold timer after one round of transmission ends, that is when the receiver sends out an ACK and the sender receives that ACK. If the node remains idle and no packet activities during $T_{thresh}$ time, it will also turn off the radio. If in the threshold time, a packet activity happens, the node will cease the previous timer and restart a new threshold timer after that packet activity completes. Another reason for applying threshold timer is to avoid unnecessarily frequent waking up and sleeping, because the time and power to turn on the radio and turn off the radio are not always negligible. Based on the statistical probability that a new packet will arrive shortly after the previous one, the node will keep idle for $T_{thresh}$ time to make sure that a possible arriving packet is not missed.

3.2 Scheduled Wakeup

In the wakeup protocol, we try to preserve the integrity of the standard 802.11 MAC protocol; meanwhile, we also hope to make a good balance among various performance parameters to get lower energy consumption, short packet latency and low packet drop rate. From the previous description, we know that an RTS-Tone can inform all the surrounding neighbor nodes that the channel will be busy in its next $L$ rounds of transmission period, thus decreasing the probability of collisions that cause unnecessary retransmissions of RTS or DATA packets. But on the other hand, it is also obvious that the RTS-Tone increases the energy consumption of the sender node, and when a collision happens, the cost will be even higher because another RTS-Tone will be sent out. To illustrate, we continue the scenario above. If node $B$ is receiving data from another node, say, $C$, this means that node $B$ is not in the idle state while the sender $A$ tries to start the transmission. Under this situation, node $A$ is out of the transmission range of node $C$, and when node $B$ sends CTS back to node $C$, suppose node $A$ happens to be sleeping. Thus $A$ doesn’t realize the channel is busy, a collision will happen on node $B$ when both
transmissions are started, and consequently, node $B$ won’t send back CTS to node $A$ and $A$ will re-send an RTS-Tone after timeout. To avoid costly RTS-Tone, we make a schedule between the sender and the receiver. Once a receiver sends an ACK and the sender successfully receives this ACK, that indicates there is a completed transmission between the sender and the receiver. Both of them will then make an appointment to simultaneously turn on the radio $T_{\text{schedule}}$ time later. $T_{\text{schedule}}$ is dynamic parameter based on the data rate, and it is encapsulated in the DATA packet. This idea above is from Matthew (2005).

According to Matthew (2005), the estimated $T_{\text{schedule}}$ is computed by the equation:

$$t_{\text{est}} = \rho t_{\text{est}} + (1 - \rho) t_{\text{diff}}$$

$$T_{\text{schedule}} = \gamma t_{\text{est}} L$$

where $t_{\text{diff}}$ is the most recent sample of packet interarrival time, $\rho = 0.9$, $\gamma = 0.1253$. The minimal value of $T_{\text{schedule}}$ is $T_{\text{min}}$.

If the sender’s queue contains packets to be sent out at that scheduled time, the sender just sends one RTS instead of an RTS-Tone before each DATA packet. If no packets are to be transmitted during the scheduled time, after $T_{\text{break}}$, both sender and receiver will turn off the radio and go to sleep. If new packets continuously arrive at the sender’s queue when an existing scheduled wakeup is running, all the following packets will be treated as a scheduled wakeup until the queue is empty. On the other hand, if a transmission is invoked by a threshold wakeup, this threshold wakeup will send exactly the first $L$ packets in the queue, the later packets will be sent by another threshold wakeup or by a scheduled wakeup. The first wakeup for a sender is always a threshold wakeup.

If a packet does arrive at a scheduled time, but transmission fails due to collisions, then the next appointment schedule between the sender and the receiver will be broken up, the next scheduled time will be set as infinity by the sender, that is also to say, the next transmission will be reset as a threshold wakeup by the sender. But the receiver has no idea that the schedule has been changed, it will continue to repeatedly wake up at the previously scheduled time. Only until a successful transmission is completed, will a new scheduled wakeup be built up again between the sender and the receiver.

4 Simulation Results

Our protocol was implemented in ns-2. To avoid energy consumption for routing, we use NOAH (NO Ad-Hoc Routing Agent) ns-2 (2004) routing protocol, and routing paths are fixed in the TCL scripts. We follow the energy parameters of Mica2 Mote sensors, the sending power $P_{\text{send}}$ is 81 mW, the receiving power $P_{\text{recv}}$ and idle power $P_{\text{idle}}$ are both 30 mW, and the sleep power $P_{\text{sleep}}$ is 0.003 mW. We also consider the time and the energy consumed during node turning on or turning off, the time from sleep state to idle state $T_{\text{trans, on}}$ is 2.45 ms, the time from idle state to sleep state $T_{\text{trans, off}}$ is 0.25 ms, and the power $P_{\text{trans, on}}$ and $P_{\text{trans, off}}$ are both 30 mW. The transmission range for each node is 152.4m (500 ft). All other parameters are listed in Figure 1.
In our simulation, we designed four scenarios to compare our performance in energy consumption, packet latency and packet drop rate with that of the protocol in Matthew (2005), where energy consumption is represented by the average energy consumed by each successfully transmitted bit, packet latency is the average latency from the sender to the receiver for all unsuccessfully transmitted packets, packet drop rate is the ratio of successfully transmitted packets to all packet arriving at sender’s queue. Each result is the average of 20 runs.

4.1 Single Hop and Single Flow Performance

Eight nodes are randomly placed in an area of 100m × 100m, Because the sensor range is 152.4m, each node can sense one another. A sender and a receiver are randomly chosen from them. All senders send packets with Poisson traffic at rate $R$. $R$ varies as 0.2, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 packets per second. The overall simulation time is 200s.

From the simulation result in Fig. 5, we see that both the energy consumption and packet latency of RTS-Tone protocol are less than STEM-Tone with increasing packet rates. There is no packet collision in this topology, but the packet drop rate is even higher when the data rate is small, this is because smaller data rate causes larger triggered wakeup interval, and sometimes the last packet cannot be sent out because the number of packets in the queue is less than the threshold, and the simulation ends before the next triggered wakeup.

4.2 Multiple Hop and Single Flow Performance

In the multiple hop and single flow scenario, there are five backbone nodes $A$, $B$, $C$, $D$ and $E$. Traffic is sent from $A$ to $E$. Only two adjacent nodes can communicate with each other directly. Each backbone node has seven neighbors. Node $A$ has six neighbors besides backbone node $B$, node $B$ node has five neighbors besides $A$ and $C$, and so on. All the non-backbone nodes are not on the path of routing and don’t send packets either. The non-backbone neighbors of one specific backbone node can only receive signals from this backbone node, and they are out of the transmission range of other backbone nodes, as in Fig. 6. The overall simulation time is 200s and the data rate varies from 0.2 to 3.0 packets per second.

An internal backbone node could be a sender or a receiver. If two adjacent nodes are sending an RTS-Tone concurrently, a longtime collision will happen because the duration of an RTS-Tone is much longer than a normal packet and the sender won’t cease the RTS-Tone even if collisions happen. With the increase of the data rate, the probability of RTS-Tone collision will also increase and the contention becomes severe which also increases drop rate. In our protocol, if a transmission fails, an RTS-Tone will be sent repeatedly until success or until the MAX-retransmit threshold is reached, thus inevitably increasing the average packet latency. From Fig. 7, We notice this side effect on packet latency is more obvious in our protocol than Matthew (2005).
4.3 Single Hop and Multiple Flow Performance

- **Multiple Senders**
  
  In this scenario, we randomly place 16 nodes in an area of 100m $\times$ 100m, and one of them acts as a receiver, all the senders send packets to this receiver. The number of senders varies from 1 to 10 but the overall number of nodes remains constant at 16. All senders are labeled, starting from 1. The senders with an odd label have a data rate of 1.0 packet/sec and the senders with an even label have a data rate of 0.2 packet/sec. Thus there are two different rates in this scenario. The overall simulation time is 500 seconds.

  This scenario is different from the previous one because the receiver has multiple triggered wakeup schedulers with different senders. All these schedulers are separated and could be overlapped. If an overlapping happens, then the multiple transmissions between different senders and this receiver have to compete. From Fig. 8, we noticed that the energy consumption and packet latency in the two protocols are similar, but in RTS-Tone protocol, the packet drop rate is around one half of STEM-Tone’s.

- **Multiple Receivers**
  
  In this scenario, we use the same topology and parameters as the previous one except that there is only one sender and the number of the receivers varies from 1 to 10. In each threshold wakeup, the sender could inform $L$ destination receivers using one RTS-Tone. The sender also has multiple triggered wakeup schedulers with different receivers, but the data rate of the sender is a cumulative one for all flows, that is to say, the packet interarrival time for multiple destinations is less than that of only one destination, thus the receivers will be woken up more frequently. From Fig. 9, we notice that both energy consumption and packet latency have better performance, and the drop rate remains almost constant with the increase of the number of the receivers.

5 Conclusions

In this paper, we presented a new energy saving MAC protocol based on STEM-Tone mechanism. We replaced a common STEM-Tone by an RTS-Tone which is composed of multiple RTSs to inform the destination receivers. Compared with STEM-Tone protocol in Matthew (2005), in our protocol, we just use one radio and one channel, and a receiver node keeps awake only if there is a incoming packet for it, otherwise it will go to sleep as much as possible. According to the simulation results, energy consumption, packet latency and packet drop rate all have a better performance in our protocol.

References


IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999


Figure 1  STEM-Beacon

Figure 2  STEM-Tone

Figure 3  Multiple RTS ($K = 6$)
Figure 4  Threshold Wakeup Protocol (K=6)
Figure 5  Single hop single flow scenario
Figure 6  Multiple hop single flow topology

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Layer Header (PLCP)</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Network Layer Header (IP)</td>
<td>20 bytes</td>
</tr>
<tr>
<td>MAC Layer Header (MAC)</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Data Size (DATA)</td>
<td>30 bytes</td>
</tr>
<tr>
<td>Bytes in each Data Packet</td>
<td>DATA+MAC+PLCP+IP</td>
</tr>
<tr>
<td>Bytes in RTS Packet</td>
<td>33+PLCP</td>
</tr>
<tr>
<td>Bytes in CTS Packet</td>
<td>14+PLCP</td>
</tr>
<tr>
<td>Bytes in ACK Packet</td>
<td>14+PLCP</td>
</tr>
<tr>
<td>Bitrate</td>
<td>40 kbps</td>
</tr>
<tr>
<td>L (Queue Threshold)</td>
<td>2</td>
</tr>
<tr>
<td>K (RTSs in RTS-Tone)</td>
<td>20</td>
</tr>
<tr>
<td>$T_{\text{listen}}$</td>
<td>1 ms</td>
</tr>
<tr>
<td>$T_{\text{sleep}}$</td>
<td>20 ms</td>
</tr>
<tr>
<td>$T_{\text{thresh}}$</td>
<td>20 ms</td>
</tr>
<tr>
<td>$T_{\text{min}}$</td>
<td>50 ms</td>
</tr>
<tr>
<td>$T_{\text{DIFS}}$</td>
<td>50 $\mu$s</td>
</tr>
<tr>
<td>$T_{\text{SIFS}}$</td>
<td>10 $\mu$s</td>
</tr>
<tr>
<td>$T_{\text{prop}}$</td>
<td>2 $\mu$s</td>
</tr>
<tr>
<td>$T_{\text{trans.on}}$</td>
<td>2.45 ms</td>
</tr>
<tr>
<td>$T_{\text{trans.off}}$</td>
<td>0.25 ms</td>
</tr>
<tr>
<td>$P_{\text{send}}$</td>
<td>81 mW</td>
</tr>
<tr>
<td>$P_{\text{recv}}$</td>
<td>30 mW</td>
</tr>
<tr>
<td>$P_{\text{idle}}$</td>
<td>30 mW</td>
</tr>
<tr>
<td>$P_{\text{sleep}}$</td>
<td>0.003 mW</td>
</tr>
<tr>
<td>Sensor range</td>
<td>152.4 m</td>
</tr>
</tbody>
</table>
Figure 7  Multiple hop and single flow scenario
Figure 8 Multiple sender scenario: one receiver receiving packets from multiple senders in a single hop.
(a) Energy consumption

(b) Latency

(c) Packet drop rate

Figure 9 Multiple receivers scenario: one sender sending packets to multiple receivers in a single hop