Data Collecting Protocol Based on a Rooted Tree for Wireless Ad Hoc Networks

SATORU OHTA
Department of Information Systems Engineering, Faculty of Engineering
Toyama Prefectural University
5180 Kurokawa, Imizu-shi, Toyama 939-0398
JAPAN
ohta@pu-toyama.ac.jp https://www.researchgate.net/profile/Satoru_Ohta

Abstract: - A simple communication protocol that effectively enables a base station to collect data from nodes in wireless ad hoc networks is proposed. The proposed protocol assumes that the network is constructed using a data transmission modem that does not support network functions; moreover, the network is assumed to be multi-hop. Because radio waves transmitted from nodes may generate interference, packets issued simultaneously will collide. Thus, the protocol must provide a routing function and avoid packet collisions. To satisfy these requirements, the proposed method employs a token passing method that utilizes a rooted tree structure, which is extracted from a given network. By utilizing the tree structure, the routing mechanism is significantly simplified. Furthermore, packet collisions are avoided by allowing only one node to transmit a packet. The algorithm that extracts a rooted tree from a given network is also shown. The feasibility and performance of the proposed protocol was examined on a prototype network.

Key-Words: - ad hoc network, tree, routing, token passing, communication protocol, wireless network

1 Introduction

Wireless ad hoc networks are extensively used for various applications [1–3]. The technical requirements for ad hoc network implementation can differ greatly depending on the application. Such differences can include wireless transmission distance, bandwidth, and node mobility. Thus, the wireless transmission system and communication protocol must be selected optimally according to the target application requirements.

Some wireless systems, such as the IEEE 802.11 series (wireless LANs) and IEEE 802.15 (ZigBee), are standardized and can be used to build ad hoc networks without modification; however, these methods are not always optimal for a particular application, and other wireless systems may be more appropriate for some applications. As an alternative system, wireless data transmission modems, which are attractive as data links for some applications because they provide robust transmission over long distances, are commercially available. Unfortunately, the modems are often designed for point-to-point transmission; thus, developing a network protocol to apply them to ad hoc networks is necessary.

Here, we consider a backbone network for a climber surveillance system, which is being developed to ensure mountain climber safety. The network connects mountain huts and enables base stations to collect climber position data. For this application, a network protocol is required to send data stored at the huts to the base station. We plan to use a 429-MHz wireless data transmission modem [4] for data links because of its low radio wave attenuation against obstacles and rainfall. However, this wireless modem was designed for point-to-point data transmission and it does not provide network functions. Thus, to use this modem in the mountain hut network, we must develop a network protocol that provides routing and avoids packet collision.

We propose a simple data collecting protocol that satisfies these requirements. The proposed protocol enables a base station to collect data from every node in a mesh, multi-hop network by executing a simple algorithm at each node. The algorithm does not require each node to manage the global network information; moreover, a node only has to know very partial neighbour node information. This protocol is based on a token passing approach [5–7], and every packet is transmitted over a rooted tree embedded in the mesh topology network. By employing the token approach and the rooted tree, packet collision is completely avoided and the routing mechanism is significantly simplified. The feasibility of the protocol was confirmed using a prototype network with nodes implemented using a small computer board and wireless data transmission modem. Moreover, the performance of the prototype system was evaluated.

The remainder of this paper is organized as follows. In Section 2, we explore the target application and describe the technical problems. In Section 3, the proposed protocol is explained. In Section 4, the prototype implementation is presented. In Section 5, experimental results are reported and conclusions are presented in Section 6.

2 Problem Description

The target application is the ad hoc network for the climber surveillance system, which is shown in Fig. 1. The system collects and manages position information of climbers in mountainous terrains. The climber position is identified using a GPS (Global Positioning System) terminal that is carried by each climber. The terminal has a wireless data transmitter that sends position information to neighbouring mountain huts. A communication node is placed at each hut to form a network and position data collected at each node is transmitted to a base station via this network. Consolidated position data facilitates the discovery and rescue of missing or injured climbers. The scope of this paper is to develop a network that connects mountain huts.

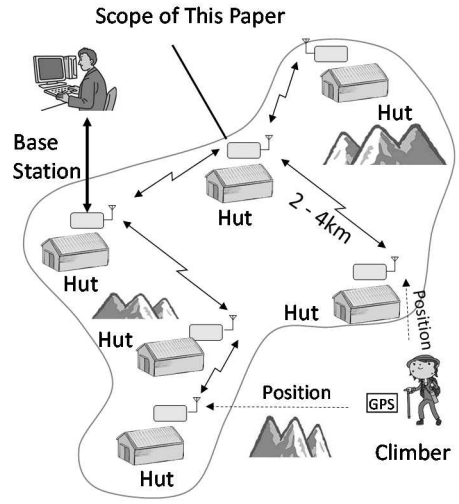


Fig. 1 Target climber surveillance system.

For the mountain hut network, building telecommunication infrastructure such as cables is not feasible because of construction difficulties, economic constraints, and environmental considerations. Thus, communication should be provided by a wireless ad hoc network.

Because of the distance between huts and expected radio wave attenuation, employing IEEE 802.11 or 802.15 wireless systems is considered impractical because the 2.4 or 5 GHz radio waves used in these systems demonstrate larger attenuation against rainfall or obstacles. To establish a reliable link, the 429-MHz data transmission modem [4] is more suitable because the 429 MHz frequency demonstrates less attenuation against obstacles and adverse weather conditions.

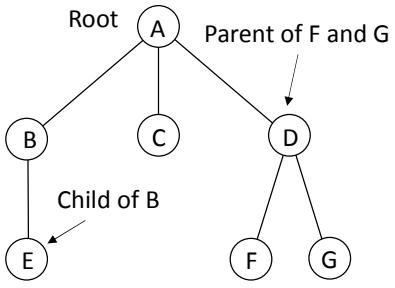
Although the data transmission modem is attractive for building the mountain hut network, the modem does not provide network functions. Thus, developing a protocol that can collect data from the huts and transmit the data to the base station is necessary. The data collecting protocol must satisfy the following technical requirements.

- Multi-hop packet transmission is mandatory because the area is too large to be covered by the wireless range of the modem.
- A link can be established between two nodes if the distance is not greater than the wireless range. Thus, the network has a general mesh topology.
- Radio waves generated from different sender nodes may interfere with each other. Thus, possible packet collision must be avoided.

For the first and second requirements, the protocol must provide a routing mechanism. Packet collision must be also avoided for the third requirement. The aim is to establish a network protocol that provides these mechanisms using point-to-point data transmission using a modem.

3. Proposed Protocol

Routing over a network becomes very simple when the network topology is a tree because the path between the source and destination is uniquely determined. In particular, if a base station is placed in the network, the rooted tree topology can be constructed by assigning the root to the base station. The rooted tree is a tree structure that comprises a root and the parent-child relationship, as shown in Fig. 2.



Fig, 2. Rooted tree.

In this topology, it is very easy for the root to collect data from other nodes. It is enough that each

node simply sends data to its parent node. Then, the parent node transmits the received data to its parent. By repeating this process, data will arrive at the root node. With this procedure, the information required at each node is its parent node. To request data collection from the root to other nodes, each node only needs to send request packets to its children. Thus, to collect data, it is sufficient for a node to know its parent and child nodes, and global network information is not required, which simplifies the node process and implementation. Because of this simplicity, the proposed protocol utilizes a rooted tree topology.

To avoid packet collision, this study investigates a protocol based on the token passing method [5–7]. In this method, only the node that has the token can transmit packets; therefore, packet collision never occurs. When the node with the token completes the data transmission, the token is delivered to the next node. To implement this mechanism, the order in which the token is passed must be determined. If the network topology is a rooted tree, token passing order is determined easily using a tree search algorithm such as depth-first search [8]. Fig. 3 shows how the token is circulated in depth-first search order.

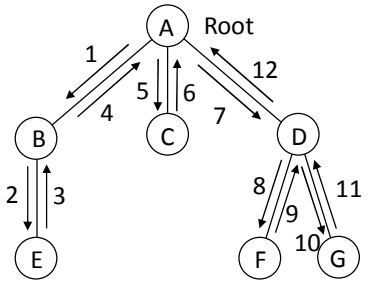


Fig. 3. Token passing in depth-first search order.

As described above, the rooted tree topology simplifies the routing for data collection as well as the token passing for collision avoidance; however, the network topology in the target application is a mesh rather than a tree. Fortunately, constructing a rooted tree for a mesh network by selecting a subset of links in the network is easy. As shown later, this selection is performed using a simple algorithm.

Based on the above considerations, this study proposes a tree-based token protocol for data collection. The protocol relies on three types of messages: *request*, *response*, and *token*. The *request* message is transmitted from a node to its children. This message is used to order each child to send data stored at the child node. The *response* message is sent from a node to its parent. The data stored at the node and its descendants is transmitted by this message. The *token* is passed after the transmission of *request* or *response* messages is completed. This shows that the sender node does not have more messages to send, and the right to transmit packets moves to the receiver node. A node is allowed to transmit packets during the period from *token* receipt to *token* transmission. Only one *token* message circulates in the network; thus, packet collision is strictly avoided.

The protocol initiates when the root node sends a *request* message. Then, if a node other than the root receives a *request* with the *token* from its parent, the node executes an algorithm to collect and send the data stored at the node and its descendants. Let n denote a node in the network. The algorithm is described as follows.

Algorithm *Data_Collection*

Given: list L_n, the children of node n;
1. **if** n is not the root
   then
2. p:= the source of *request*; // parent of n
3. Send *response* including the data stored at n to p;
   else
4. Send the data stored at n to the application;
   end if
5. **for each** c in L_n **do**
6. Forward the *request* and *token* to c;
7. Receive the *responses* from c and put them into the buffer;
8. Wait for the *token* to be returned from c;
   end for
   if n is not the root
   then
9. Forward the *responses* in the buffer to p;
10. Return the *token* to p;
    else
11. Send the *responses* in the buffer to the application, and wait for the next command from the application;
    end if

With this scheme, the *token* message is first sent from the root node and then circulated in the network. Evidently, from the algorithm, the *token* arrives at every node in depth-first search order. If a node receives a *request*, its local data is returned to the parent (line 3). Furthermore, data stored in its descendants is collected (lines 6–7) and forwarded to the parent (line 9). By performing these steps at every node recursively, the root can collect the data stored at every node.

As seen above, global network information is not required at each node. The node only has to know the list of its child nodes, i.e., very partial information on the rooted tree, which simplifies the process and

implementation of the protocol. The children list L_n can be obtained by the following method.

To determine a rooted tree in the mesh topology network and obtain the children list L_n for node n, the proposed method employs a "*tree maker*" message, which is issued by the root and forwarded from a node to its child node. The *tree maker* message includes a list of nodes that have not been included in the tree. Let U denote this list. If a node, which is denoted by n, receives the *tree maker*, it deletes itself from U. This occurs because n is now connected to the tree. Then, the node looks for its children among the elements in U. If some nodes in U are found to be reachable from n, they are added to the children of n. Then, n sends the *tree maker* to its newly discovered children, and the process is executed recursively. Finally, n returns the updated U to its parent.

This procedure adds reachable nodes to the tree in a greedy manner until all nodes are included. When a node is added to a tree, it is deleted from U. The deleted node will not be checked again as a child candidate of other nodes; thus, a cycle will not occur in the resultant topology. In other words, the obtained topology is assured to be a tree. This computation is summarized as follows.

Algorithm *Build_Tree*

```
1.  Ln:= {}; // Children of n
2.  if n is the root
3.      then U:= {all nodes}
        else
4.          U:= node list given by the tree maker;
5.          p:= the source of the tree maker; // Parent
    end if
6.  U:= U – {n};
7.  for each c in U do
8.      if c is reachable from n then
9.          Ln:= Ln ∪ {c};
10.         U:= U – {c};
        end if
    end for
11. for each c in Ln do
12.     Send tree maker to c with node list U;
13.     U:= node list replied from c;
    end for
14. if n is not the root then reply U to p;
```

Fig. 4 shows an example of a tree found by the above algorithm. The thick lines are the links used for the tree, and the dotted lines are the links that are not used for the tree. The algorithm starts at the root node A. Initially, U is set to {A, B, C, D, E, F, G} (line 3). Since node A is the root and is included in the tree, A is deleted from U (line 6). Then, lines 7–10 find that nodes B and C among {B, C, D, E, F, G} are reachable from A. Thus, children list L_A will be set to {B, C}. Next, the algorithm is executed at B and C (lines 11–13). By executing the algorithm at B, it finds that D, E, and F are reachable from B. Thus, D, E, and F become the children of B, and B returns {C, G}. This list is sent to C and the algorithm is executed at C. Then, G, which is reachable from C, is added to the tree. Node C will reply with an idle list to root A. After this procedure is completed, the children lists L_A,…, L_G are correctly set up.

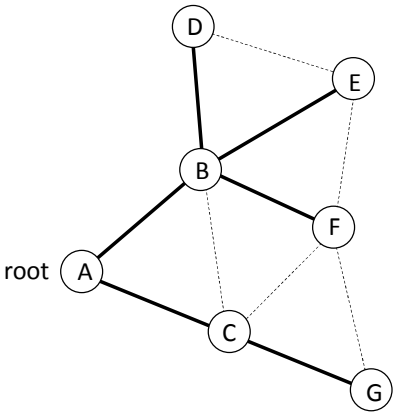


Fig. 4. Tree discovered by algorithm Build_Tree.

4. Prototype Implementation

A prototype ad hoc network was implemented using the proposed protocol. The feasibility of the proposed protocol was confirmed using the prototype. This section explores the hardware and software of the implemented prototype.

4.1 Hardware

Each node of the network was constructed by connecting a wireless data transmission modem and a computer board. The computer board is a Raspberry Pi 2 Model B [9], which has a BCM2836 processor and 1 GB RAM. The Linux Raspbian distribution was installed as the OS. The computer exchanges commands and data with the modem through a universal asynchronous receiver transmitter (UART) serial interface.

This prototype included a Circuit Design Inc. MU-1N wireless modem [4]. This modem provides considerably stable wireless communication by utilizing a 429-MHz radio wave. The proposed protocol was implemented using the following modem commands.

- Data transmission (sends data to the specified destination)

- Packet transmission check (tests whether the destination responds)
- Received signal strength indication (RSSI) measurement (estimates the signal and noise levels at the specified destination node)
- Identifier setting
- Carrier sense

Each modem command is performed by sending a particular character sequence, e.g., @DT for data transmission, from the computer through the UART interface. The data size for the data transmission command is limited to 255 Bytes. The bit rate of the wireless link is 4800 bps; thus, the modem link is relatively narrow-band. However, the current target application does not require higher bit rate, and the modem specifications are adequate for this application.

The prototype node is shown in Fig. 5.

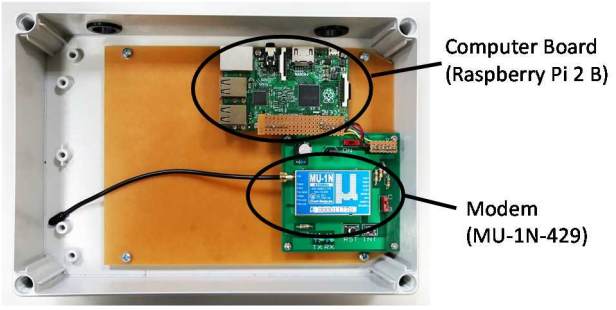


Fig. 5. Prototype node.

4.2 Software

The proposed data collecting protocol was implemented as two programs that run on the computer board. One program is performed on the root node. This program waits for command input by the operator. If the program receives a command, the data-collecting algorithm is initiated. The other program is executed on nodes other than the root. This program starts the algorithm if the node receives a *request* message from its parent node. These programs were written in Perl with optional packages to control serial communication and the BCM2836 processor.

Request, *response*, and *token* messages are exchanged by the data transmission command provided by the modem. The packet structure of these messages is shown in Fig. 6. The first field of the packet is the message type, i.e., *request*, *response*, or *token*. The next field is option length, followed by option field. Currently, the option field is used to indicate the identifier of the source node. The fourth and fifth fields are used for the application. The fourth field identifies the application level command, whereas the fifth field includes the arguments given to the command. The algorithms shown in Section 3, i.e., *Data_Collection* and *Build_Tree*, were implemented using this packet format.

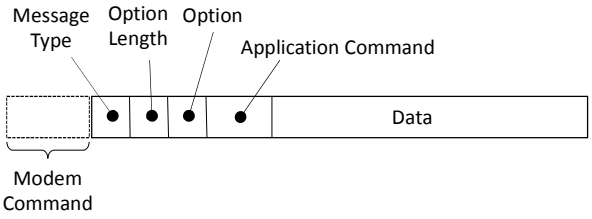


Fig. 6. Packet format.

The application level commands were developed for two purposes. The first purpose is to demonstrate the feasibility of the proposed protocol for the mountain hut network application. From this perspective, we implemented several commands that allow the root node to collect the data stored in each node. Second, to operate the network in real-world applications, network management functions are indispensable; thus, some basic management commands were developed.

The application level commands include *get*, *watch*, *update*, *copy*, *ping*, *rssi*, *build*, and *showtree*. The *get* command collects and shows the content of a specified text file stored at each node. The *watch* and *update* commands are used together. The *update* command dumps the latest addition to the file being monitored at each node. The monitored file is specified through the *watch* command. The *copy* command fetches a file from a remote node and stores a copy at the root node. The feasibility of the mountain hut network can be confirmed using these commands.

Other commands are provided for management purposes. The *ping* command confirms that all nodes in the network are alive. For this command, the packet transmission check function of the modem is used to see whether a child node is reachable. The result is reported to the root node by *response* messages. The wireless link performance is checked using the *rssi* command. When this command is initiated, each node measures the signal and noise levels for its parent and children nodes using the modem's RSSI function. The result is notified to the root as a *response* message. The *build* command constructs the tree structure by executing the *Build_Tree* algorithm at each node. The current tree structure can be traced using the *showtree* command, which lists the child nodes of each node in the network.

5. Experimental Results

The proposed method was examined through experiments using a small network that consisted of four prototype nodes. First, the tree construction algorithm was examined using two different node placements in our laboratory. These placements are shown in Fig. 7. In the placement shown in Fig. 7 (a), only node B is reachable from the root node. The other nodes are unreachable because of radio wave attenuation caused by walls. Meanwhile, node C is out of the wireless range of the root node A in the placement shown in Fig. 7 (b).

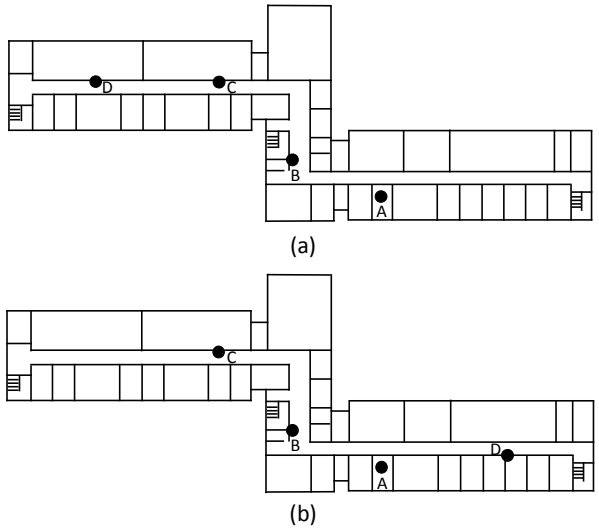


Fig. 7. Floor map and experimental node placement.

For these node placements, the *build* command was executed, and the obtained tree was extracted by the *showtree* command. Consequently, the trees shown in Fig. 8 were discovered. Fig. 8 (a) and (b) show the trees found for the placements shown in Fig. 7 (a) and (b), respectively. For both cases, the trees were constructed adequately with two nodes connecting within reachable distance. From this result, it is confirmed that the tree construction algorithm works successfully.

For the trees constructed using the build command, all application level commands worked correctly. For example, the output of the *rssi* command is shown in Fig. 9.

Note that the response time varies depending on the command. The response for the *showtree* command is completed within 1.9 s. The *ping* command takes longer, i.e., 2.6 s because each node issues test packets to its children and waits for the replies. Note that the *build* command took 10 s. The response of the *build* command is slower because a timeout mechanism is used to verify node reachability.

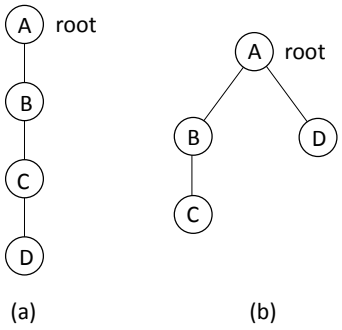


Fig. 8. Obtained trees for (a) node placement 1 and (b) node placement 2.

```
Node 31 -> Node 33
S: -75 dBm, N: -119 dBm

Node 33 -> Node 31
S: -72 dBm, N: -113 dBm

Node 33 -> Node 47
S: -83 dBm, N: -117 dBm

Node 47 -> Node 33
S: -84 dBm, N: -121 dBm

Node 47 -> Node 48
S: -70 dBm, N: -113 dBm

Node 48 -> Node 47
S: -70 dBm, N: -116 dBm

-- response completed --
```

Fig. 9. Results of executing the rssi command.

For the data collecting application, assessing the data transmission performance is important. Therefore, the time taken to complete the *copy* command was measured to evaluate this performance. For this purpose, three files of different sizes were stored on the microSD card of the computer. Then, the *copy* command was executed to transmit these files from a node to the root, and the elapsed time from the command input to the end of transmission was measured. The speed of reading data from the microSD card is much higher than the modem's bit rate; thus, its effect on performance is negligible. The file sizes were 10^3, 10^4, and 10^5 Bytes.

Fig. 10 shows the response time of the *copy* command against file size for single hop transmission. As shown in Fig. 10, the response time increases nearly proportionally with increased file size. From the increase rate against transmitted data size, the estimated bit rate is approximately 3500 bps. Thus, the obtained bit rate is less than the modem speed (4800 bps) because some overhead process exists for sending and receiving a packet in the modem.

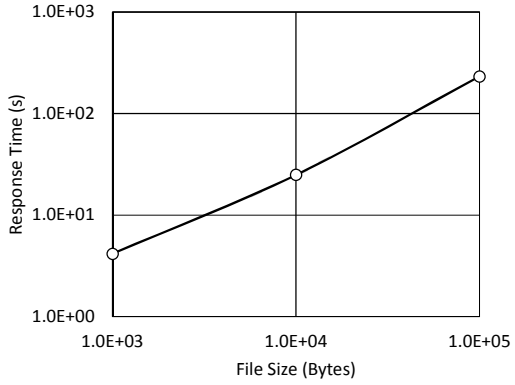


Fig. 10. Response time versus file size for single hop transmission.

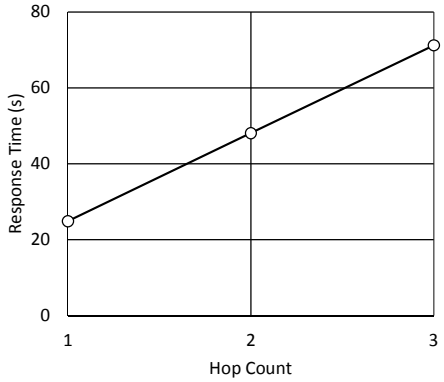


Fig. 11. Relationship between response time and hop count.

Moreover, overhead exists in the developed software that runs on the computer board. For example, the software checks the response from the child node before sending a *request* message. Furthermore, before every packet transmission, carrier sense is executed to assure the absence of other transmissions. Currently, these processes are performed to ensure reliable communication. However, the necessity of the response check and carrier sense is currently uncertain. If the overhead processes can be omitted, the response time and bit rate can be improved.

Fig. 11 shows the relationship between hop count and the response time of the *copy* command for a 10^4 Byte file. Fig. 11 shows that the response time increases linearly with increased hop count. This characteristic is obtained because data cannot be transmitted simultaneously from two or more nodes using the token mechanism. Suppose that it takes T seconds for a node to send data for a single hop. Then, for three-hop transmission, the origin and two transit nodes require T seconds to send the same data and the transmissions cannot be performed simultaneously. Obviously, this requires a data transfer time of $3T$ seconds. This characteristic may be a limitation for delay-sensitive applications.

6. Conclusion

A simple data collecting protocol has been proposed for a wireless ad hoc network. The proposed protocol provides packet collision avoidance and routing functions based on a token passing approach and a rooted tree topology. We have demonstrated how a rooted tree topology is extracted from a mesh network with a simple algorithm. The feasibility of the proposed method was confirmed with a prototype, which included a computer board and a data transmission modem.

References:

[1] C.-K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall, 2002.

[2] S. Misra, I. Woungang, and S. C. Misra, *Guide to Wireless Ad Hoc Networks*, Springer, 2010.

[3] M. Frodigh, P, Johansson, and P. Larsson, "Wireless ad hoc networking - the art of networking without a network," *Ericsson Review*, 77, 4, pp. 248–263, April 2000.

[4] Circuit Design Inc., *MU-1N-429 Manual*, http://www.circuitdesign.jp/jp/products/products2/doc/MU-1N-429.pdf (in Japanese).

[5] W. Stallings, "Local network performance," *IEEE Communications Magazine*, 22, 2, pp.27–36, Feb. 1984.

[6] *Token-Passing Bus Access Method*, ANSI/IEEE Standard 802.4, 1985.

[7] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," *in proc. the 2nd European Workshop on Wireless Sensor Networks*, pp. 108–120, Istanbul, Turkey, Jan. 2005.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, I*ntroduction to Algorithms, Third Edition*, The MIT Press, 2009.

[9] *Raspberry Pi*, https://www.raspberrypi.org/, 2015.