Electronic Preprint for Journal of Information Processing Vol.24 No.2

Regular Paper

A Practical and Efficient Overhearing Strategy for Reliable Content Distribution over a Single Hop Ad Hoc Network

Hiroaki Yokose[1] Koji Nitta[1] Satoshi Ohzahata[1,a)] Toshihiko Kato[1]

Received: June 29, 2015, Accepted: December 7, 2015

Abstract: Nowadays, wireless LAN is equipped in most portable devices and widely deployed, and then we can communicate over wireless link with low cost. However, when the same content is simultaneously downloaded by several terminals with unicast in a wireless LAN, the bandwidth must be divided among the terminals and the download speed is reduced. With using the broadcast nature of wireless media, pseudo multicasting methods, which overhear the unicast flow to realize multicast like communication, have been proposed. In the pseudo multicasting methods, since parts of the content is obtained by overhearing the TCP/UDP flows of a specific content going to other terminals in a wireless LAN, and the server need not send the content for each client repeatedly. These methods enable to realize multicasting with a practical way. However, redundant unicast traffic is still generated because the receivers do not obtain contents via overhearing in an efficient way. Based on the pseudo multicasting, we propose an efficient, flexible and reliable content distribution method over single hop wireless ad hoc network in this paper. In the proposed method a content is divided into pieces and the server distributes it according to the download status of each receivers at the application level. To accelerate downloading, we propose a method for selecting the piece and the terminal for transmission taking the effect of network coding for efficient overhearing. We developed a testbed and confirmed that the proposed scheduling accelerates the download speed even when the communication speed of each terminal is different.

Keywords: ad hoc network, content distribution, overhearing

1. Introduction

Communication speed in wireless LAN has been increasing each year, and a large file is often shared among tablet devices over wireless LAN. However, when many terminals simultaneously intend to download a specific large file via unicasting in a wireless LAN, the download speed is reduced even though the file is being downloaded via an identical wireless network. This is because the bandwidth of the broadcast media has to be shared among the terminals. In addition, since the same content is repeatedly downloaded, the bandwidth is consumed by the redundant traffic. For these reasons, a method for accelerating the distribution of content is required for a wireless LAN.

When a specific content is distributed to many terminals by unicast, the content has to be distributed multiple times. To reduce such redundant traffic, the reliable multicasting [1], [2] is proposed for wireless networks. However, when wireless conditions at the receivers are different, the transfer speed for all terminals is limited by the speed of the slowest terminal. So as not to reduce the transfer speed for the multicast terminals, in the hybrid approach, the server also sends the same data using unicast to the terminals operating under bad conditions [2]. Even though a multicasting in the transport layer has to rely on the media access control method in a wireless network, effect of MAC retransmission is not well considered.

Farther improve the problems in a practical way, pseudo multicast methods [3], [4] are proposed to solve the problems that a multicast cannot consider wireless conditions for each receiver and the retransmission control of IEEE 802.11 MAC. In the pseudo multicast, a target node is selected as a destination of unicast flow, and the other terminals downloading the same content overhear the unicast flow. Since the pseudo multicast communicate with a unicast flow, retransmission controls of MAC and TCP are effectively used and the communication speed of IEEE 802.11 physical layer can be selectable for wireless conditions of the terminal by selecting the target node adaptively. The pseudo multicast effectively works because a wireless link is error-prone and each terminal has different wireless conditions in general. However, it is difficult to effectively select the target node while considering the range of the overhearing and MAC transmission speed. In addition, the above methods rely on the sequence number of the transport layer for the reliability of the byte stream, and then all the terminals have to simultaneously start downloading at the beginning of the content distribution.

Ditto [5] and REfactor [6] apply the content piecing to a wireless network to improve the network utilization. In these methods, the overhearing of a wireless link is used to cache content pieces at the nodes in the communication route or the terminal

[1] The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[a)] ohzahata@is.uec.ac.jp

A part of this paper is presented in AINTEC 2013.

when pieces of the content are downloaded by the client terminal in the router. Then, these cached pieces can be reused when these are repeatedly requested by the clients along the route to reduce the redundant traffic in a multi-hop wireless network or a wireless LAN. The content piecing also enables to release the restriction of the byte stream communication whereby the clients have to join the multicast group from the beginning of a content download. However, since these methods do not consider a piece diffusion strategy in the wireless network to accelerate downloading, overhearing is not used efficiently.

BitTorrent [7] is a popular peer-to-peer content sharing system, and enables the distribution of large files among many terminals. In BitTorrent, a file to be downloaded is divided into small pieces in the application layer and these pieces are exchanged among the recipients. Since the parallel download is realized by the exchange of file pieces among terminals, the terminals can get the higher throughput than that achievable with download from the server. In addition, file piecing provides flexibility and reliability in downloading. The client need not download a file from the beginning, and can download a file at different timings and from different positions within the file. Since completeness of the pieces is checked by their hash value, the reliability can also be served at the application layer level. However, since BitTorrent constructs an overlay network in the application layer, the content is delivered by unicast. Then, when BitTorrent transfers a content over a wireless LAN, a collision occurs in MAC layer if multiple terminals transmit a data frame at the same time. Therefore, the terminals are not allowed to perform a simultaneous communication in the communication range under a CSMA/CA control, and the parallel downloading does not work effectively [8].

Network coding [9] improves the throughput because the packets are XOR coded from multiple packets, and the number of transferred packets is decreased. For an efficient communication in a wireless network, network coding can be added to the retransmission control of the MAC layer to reduce the number of retransmissions [10], [11]. However, when the network coding effectively works, the coded frames have to be correctly received at all receivers by broadcasting. Then, since the communication conditions in a wireless network do differ among receivers, network coding in the MAC layer for a wireless network does not effectively work in general.

In this paper, we propose a reliable content distribution method based on the pseudo multicast over single hop wireless ad hoc network[*1], whose basic idea was presented in Ref. [12]. The method is proposed to share contents (documents, photos, videos and so on.) among a group of up to a few dozen terminals via wireless LAN, and supposes that the receivers start downloading before ending the download of the other terminals to realize simultaneous downloading of a part of the content via overhearing efficiently. In the method, since the content is divided into small pieces in the application layer and is sent and managed in units composed of some pieces, terminals can asynchronously download using file piecing, and the method can be adapted easily to different wireless conditions for multiple terminals because each communication is controlled by the TCP unicast. However, we found that there were limitations to how much improvement can be achieved when each terminal needs to download different pieces, and the previous method cannot use the bandwidth of the wireless network efficiently. Then, we propose piece scheduling to allow adaptation when the wireless conditions of the terminals are different and each terminal starts downloading at a different timing, and also add network coding in the application level to the above mentioned method. The proposed method is implemented in a testbed and the evaluation results show its effectiveness.

This paper is organized in as follows. Section 2 shows an overview of the proposed method, and the proposed scheduling methods are described in Section 3. Section 4 evaluates the proposed method in testbed environments, and we discuss some issues of the proposed method in Section 5. Finally, Section 6 concludes this paper.

2. Content Distribution with File Piecing and Overhearing

2.1 Overview

In multicast or broadcast methods, a content is simultaneously transferred to many terminals with the broadcast nature of a wireless media. However, these methods cannot work effectively because conditions of wireless communication are different for each terminal. Then, pseudo multicast methods [3], [4] improve problems of multicasting by overhearing the unicast flow because a retransmission control and a transmission speed control of MAC protocol for unicast can be used. Then, the transfer speed of the physical layer can be adjusted with considering the communication quality of the receivers. However, there are still unsolved problems below.

Problem 1: The terminals cannot join content downloading at the middle of the communication of the other terminals because a byte stream is used as the data structure of the communication and a part of the content cannot be distinguished in the stream.

Problem 2: In case of downloading the same content, each terminal does not always request the same part of the content because timings of joining content downloading are different for each terminals. This means the same part of the content has to be delivered repeatedly and generates redundant traffic.

Problem 3: A pseudo multicast method employs the target node as the destination of the unicast flow. However, it is difficult to effectively select the target node because the transmission range and the transmission speed depend on communication conditions of the target node, and then the selection method affects the performance of content delivery.

To solve the problems 1 and 2, we proposed a method that uses file piecing and overhearing to produce a reliable content distribution over single hop wireless ad hoc network environments even when communication conditions at each terminal are different [12]. The proposed method is based on the pseudo multicast method, and a content (e.g., software or video) is divided into

[*1] References [2], [3], [4] suppose the infrastructure mode of wireless LAN and a server multicasts via single AP or multiple APs, but our method supposes that a content is downloaded with ad-hoc mode without infrastructure.

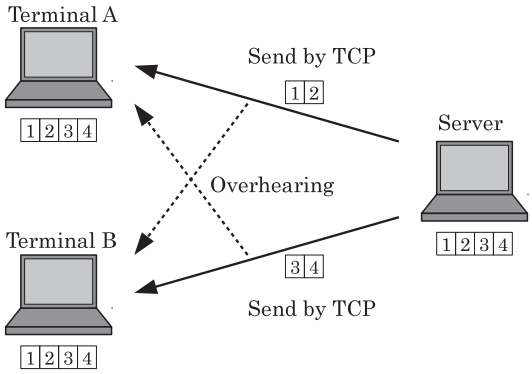


Fig. 1 Overview of the proposed method.

pieces, and sends them by TCP. The terminals receive their own pieces and also overhear and use pieces addressed to other terminals to realize the pseudo multicast (**Fig. 1**). In addition, network coding is used to accelerate file downloading. Based on them, the destination and piece selection scheduling methods are proposed to solve problem 3. Overview of the proposed method is described in this section, and the scheduling method is shown in Section 3.

2.2 Pseudo Multicast with Content Piecing

In a wireless ad hoc network environment, the communication quality at each terminal differs because of the distance or obstacles between the server and the terminals. However, the content server must control the transfer rate and the reliability so as to produce a higher throughput. In the conventional pseudo multicast method, a multicast-like communication is achieved by overhearing the unicast flow of the other terminals. Since TCP is used for the communication, we can also use the retransmission mechanism, congestion control and reliability control of TCP, and Auto Rate Fallback (ARF) and retransmission mechanism in the MAC layer as they are. However, since TCP sends a byte stream and the sequence number starts with a random number, the overheard data's position in the file cannot be specified in the transport layer, and this fact produces Problems 1–3 in Section 2.1.

In our method, a file is divided into pieces and an application-level header is added to specify the position. With using information of the content pieces, the terminals report the piece possession status to the server, and the server thereby obtains information on the piece possession, the download speed and the overhearing relations among the terminals. Using this information, the clients do not have to simultaneously start a content downloading at the beginning (solution for Problems 1 and 2). Then, the server can improve piece transfer scheduling; i.e., which piece should be addressed as the destination terminal, taking the overhearing status into consideration (a solution for Problems 2 and 3).

2.2.1 Asynchronous Download by Content Piecing

When users download a file, they start the download at different times and their communication environments are usually different in general. However, since the multicast protocol in the transport layer sends out content as a byte stream, all the terminals have to start downloading the content from its beginning at the same time.

In our method, since a file is divided into small pieces in the application layer and distributed by TCP unicast, these pieces are adequately delivered by unicast controls of each layer even for a different start timing of download and different wireless conditions for each terminal. To identify the pieces from overheard frames, pieces are sent with a header. The header includes an identifier for the piece, the length of the piece, the position of the piece in the file, and the destination. Since the terminals download a file in application level pieces, the terminals can request different pieces of it at different times. In addition, the server obtains requests for pieces and a notification of the pieces received at the terminals. Such information is useful when the server selects pieces to transfer for an efficient use of the bandwidth.

To further improve the efficiency, we apply network coding for the pieces in the application level. Since a reliable byte stream is delivered by TCP or a part of a reliable byte stream is checked by the hash map for pieces, network coding effectively works even for the overhearing terminals with different wireless conditions.

2.2.2 Reliability Control by Content Piecing

For a reliable communication, retransmission and congestion controls are implemented on the MAC or TCP layer, and these controls and the pseudo multicast can easily use these controls of the unicast. In the proposed method, the destination terminal of TCP confirms the reliability of received data, and the other overhearing terminals can check the reliability of the byte stream of overheard data by the sequence number and checksum of TCP without additional message exchanges. In addition, since the server distributes a file as pieces at the application level, all the terminals can correctly identify the byte streams received and request pieces that were not received from the server.

2.3 Network Coding for Piecing

We apply network coding [9] to improve the efficiency of our content distribution with reducing the number of retransmissions. The network coding is applied for the pieces, and XOR is calculated for each correspondent bit for pieces. In the case that the server sends pieces of P_1 and P_2 but the receiver cannot receive P_2, the server has to retransmit P_2. Then, the sender makes an XORed piece as in Eq. (1), and sends P_N with the information of the piece numbers as the header to the receiver.

$$P_1 \oplus P_2 = P_N \tag{1}$$

Then, the receiver calculates XOR to get P_2 as in Eq. (2)

$$P_N \oplus P_1 = P_2 \tag{2}$$

This method effectively works in the case that some of the receivers do not correctly receive pieces P_1 or P_2 because P_N piece will be the same effect of a retransmission control for both of the pieces P_1 and P_2 with single transmission. However, since the coded piece of P_N has to be received by all the receivers, the server sends the coded piece with the lowest transmission speed of the receivers. This control also increases a range of overhearing and effect of network coding.

2.4 Procedures of Downloading

In the proposed method, the terminal establishes a TCP con-

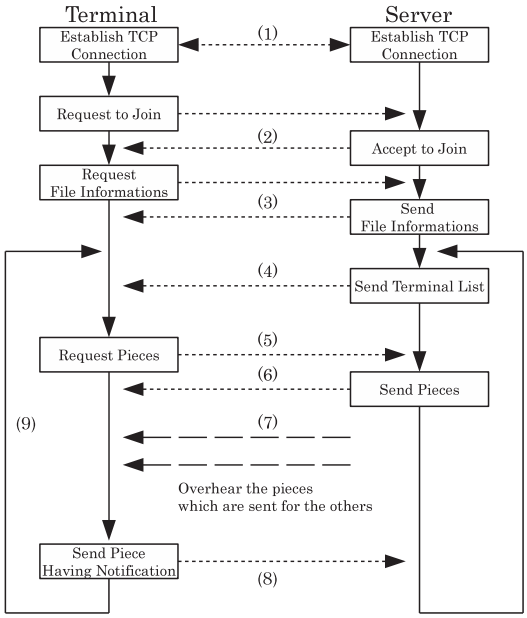


Fig. 2 Communication procedures.

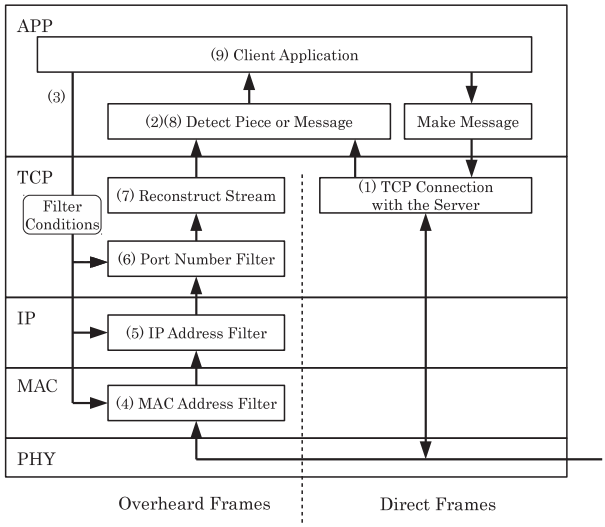


Fig. 3 Control procedures for overhearing.

nection and downloads pieces via the connection directly, and also via the overhearing. The terminals repeat the direct piece reception and the reception of overhearing until completion of the download. These procedures are described in **Fig. 2**.

(1) A TCP connection is established between the server and the terminal.

(2) The terminal requests to join the group receiving the file. If the server accepts the request, a unique ID is given to the terminal.

(3) The terminal requests information on the file including the overall length, the piece length and a hash map for the piece validation, and then the server replies to the request.

(4) The server sends a list of terminals that are joining the group, and also sends information of the new terminal (terminal ID, IP addresses, port numbers) to the other terminals.

(5) The terminal requests pieces from the server.

(6) The server decides which pieces to send and their destination according to the piece requests and the piece possession status and overhearing conditions of the terminals, and then sends the pieces. As long as no terminal joins, leaves or requests information, the server continues to send pieces.

(7) The terminal overhears the pieces using information from the server.

(8) The terminal notifies its overhearing status periodically. Since the destination address is added to each piece distribution, the overhearing terminal can get a success ratio of overhearing for each terminal in the group. Then, if the constant time is elapsed without the overhearing notification from the terminal, the server determines that the client failed to overhear (overhearing timeout).

(9) The terminal repeats requesting pieces, receiving them and notifying its status until the terminal has received all pieces of the file. When the terminal finishes downloading, the terminal leaves the group, and the server notifies this leaving to the other terminals.

2.5 Pseudo Multicast Procedures of Overhearing

In the proposed method, the terminals overhear frames for the other terminals that are downloading the same file. To realize pseudo multicast, the terminals has to obtain frames to the content in downloading. Thus, practical multicast or broadcasting is realized by overhearing unicast flows that deliver the same content ID in the application. For this reason, the server makes a group of unicast communications for the downloading content, and periodically reports the information to the receiver terminals. The receiver terminal selects frames for the content, reconstructs the TCP byte stream, and extracts the content pieces from it.

Figure 3 shows the overhearing procedures in the terminal. The terminal receives a message that includes information for overhearing from the server (Procedure (1), Procedures (1)–(3) in Fig. 2). The client application in the terminal controls filters on each layer according to the information from the server so as not to pass frames/pieces that are not needed to the upper layer (Procedure (3)–(6)). The TCP byte stream is reconstructed by the sequence number and segment length for each flow (Procedure (7)). Provided the sequence number is continuous, the reconstructed stream is correctly received without loss. If a gap is found in the overheard byte stream, interruption of the byte stream is notified to the upper layer because the terminal cannot request a retransmit by TCP. The reconstructed TCP byte stream is sometimes discontinuous. However, if the byte stream is valid, the terminal can obtain some pieces from the byte stream, and then these pieces are passed to the application (Procedure (2) and (8)). The application saves the pieces if the pieces are a part of a downloading file and have not yet been received. Then, the terminal periodically notifies its status with respect to received pieces to the server (procedure (9)).

3. Scheduling for Piece and Terminal Selection

In the pseudo multicast methods, the server selects the receiver of the unicast as the target node [3], [4]. The target node has to be selected carefully because the transfer ratio of physical layer affects the communication quality such as the transmission range, the error ratio, and the retransmission ratio. In addition, our

method divides the content into pieces when these are delivered to terminals, and some of the terminals often need the same pieces simultaneously. Then, the piece selection of the proposed method also affects the number of pieces effectively used in the terminals because the number of terminals requesting the pieces is different for each piece. The destination terminal selection and piece selection have to be decided during scheduling. For the above reasons, the server selects a piece and a terminal for sending, taking the piece request status and the piece reception status via overhearing into consideration. In the following, we show three methods for selecting a destination terminal and a piece selection when the server transfers the piece.

3.1 Method A: Simple Selection

Algorithm 1 Simple scheduling.

```
▷ k ← 0 initialization.
▷ Select a terminal in round robin manner.
T_dest ← Terminals[k % Terminals.length]
k ← k + 1
score_max ← 0
for i = 0 to T_dest.Pieces.length−1 do
    if T_dest.Pieces[i] = 'requested' then
        s ← 0                          ▷ s is # of terminals requesting piece i.
        for j = 0 to Terminals.length−1 do
            if Terminals[j].Pieces[i] = 'requested' then
                s ← s + 1
            end if
        end for
        if s > score_max then
            pid ← i                    ▷ Select piece with maximum s.
            socre_max ← s
        end if
    end if
end for
return T_dest, pid                     ▷ Destination: T_dest, Piece ID: pid
```

In method A, the destination terminal is selected in a simple round robin manner, and the method is described in Algorithm 1 [12]. The sender selects the piece which has the largest number of requests from the terminals, and the piece has to be required by the terminal selected as the destination. In this process, each terminal is assigned at least $1/N$ of the total receiving opportunities (N is the number of terminals). In addition, many terminals can obtain a requested piece through a single transmission because the piece is requested by many terminals. Note that *Terminals* is an array of terminals, the member function *Pieces* represents its member function (Requested by the terminal: 'requested', and Received by the terminal: 'have').

3.2 Method B: Considering Piece Reception Status

In cases where the success ratio of overhearing is low or the requested pieces are different, a terminal cannot get pieces from overhearing and the download speed is reduced for the terminal. In these situations, the number of pieces received in the slow terminal should be increased. Then, method B takes into account the slow terminal as in the pseudo multicast, and then the piece is selected with considering the effect of overhearing. This selection method is described in Algorithm 2 and Algorithm 3.

Algorithm 2 Considering Piece Reception Status.

```
▷ Select Terminal having the lowest E as Destination.
T_dest ← Terminals[0]
for i = 1 to Terminals.length−1 do
    if Terminals[i].E < T_dest.E then
        T_dest ← Terminals[i]
    end if
end for
▷ Calculate # of terminals expected to receive.
score_max ← 0
for i = 0 to T_dest.Pieces.length−1 do
    if T_dest.Pieces[i] = 'requested' then
        s ← 0  ▷ s is a sum of reception ratio of terminals requesting piece
i.
        for j = 0 to Terminals.length−1 do
            if i = j then
                s ← s + 1
            else if Terminals[j].Pieces[i] = 'requested' then
                s ← s + Terminals[j].R[T_dest]
            end if
        end for
        if s > score_max then
            pid ← i                    ▷ Select piece with maximum s.
            socre_max ← s
        end if
    end if
end for
return T_dest, pid                     ▷ Destination: T_dest, Piece ID: pid
```

In Algorithm 2, the ratio of the number of pieces effectively received in a terminal is defined as E_n in Eq. (3). In the equation, "# of pieces received in terminal n" also includes pieces via the overhearing.

$$E_n = \frac{(\#of\,pieces\,received\,in\,terminal\,n.)}{(\#of\,pieces\,sent\,from\,the\,server.)} \tag{3}$$

E_n is calculated in the server every time when the piece is transmitted to each terminal. The terminal which has the smallest E_n is selected as the destination of the piece. With this control, opportunities for the piece to be received become proportionally equal because the number of transmissions for the terminal is increased. In addition, a terminal which has a low success ratio of overhearing because of the wireless conditions is often selected as the destination. Since the server often selects the terminal having a low transmission rate at the physical layer, the success ratio of overhearing is also improved.

In Algorithm 2, a member variable of array E shows the number of terminals receiving a piece for a single transmission, and member variable R is the success ratio of overhearing (in case where the destination is the other terminal). First, the terminal which has the lowest E_n is selected as the destination. Next, the server lists the pieces which the destination terminal is requesting, and calculates the success ratio of overhearing in each terminal for all the listed pieces. A piece (*pid*) which has the largest number of expected receivers (Eq. (4)) is selected for the transmission. Note that $N = Terminals$.length is the number of terminals in downloading, and T_{dest} is a destination terminal in Algorithm 4.

Algorithm 3 Update of # of pieces effectively received.

▷ Piece m is transmitted to Terminal n.
for $i = 0$ to $Terminals$.length−1 **do**
  if $i = n$ **then**
    $Terminals[i].E \leftarrow \alpha \times Terminals[i].E + (1 - \alpha)$
  else if $Terminals[i].Pieces[m]$ = 'requested' **then**
    $Terminals[i].E \leftarrow$
      $\alpha \times Terminals[i].E + (1 - \alpha) \times Terminals[i].R[n]$
  else
    $Terminals[i].E \leftarrow \alpha \times Terminals[i].E$
  end if
end for

Algorithm 4 Update of Success ratio of overhearing.

▷ The server receives a having piece notification from Terminal n because of overhearing the piece addressed to Terminal m.
$Terminals[n].R[m] \leftarrow \beta \times Terminals[m].R[n] + (1 - \beta)$

▷ The server does not receives a having piece notification within the overhearing timeout.
$Terminals[n].R[m] \leftarrow \beta \times Terminals[m].R[n]$

$$(\#of terminals expected to receive the piece) = \sum_{n=0}^{N-1} \begin{cases} Terminals[n].R[T_{\mathrm{dest}}] \\ \quad : Piece is effectively used. \\ 0 \quad : Piece is not effectively used. \end{cases} \quad (4)$$

In the above procedures, the download speed is fairly controlled among the terminals for status of download, wireless link and overhearing.

Note that the number of pieces effectively received in terminal n, $Terminals[n].E$, is updated with Algorithm 3. $Terminals[n].E$ is updated every time a piece is transmitted and the value is smoothed by α ($0 \le \alpha \le 1$). Before smoothing, $Terminals[n].E$ is set as 1 for the case where the piece is sent as the destination, is set as 'the success ratio of overhearing for the terminal' for the overhearing, and is set as 0 otherwise. $Terminals[n].R[m]$ is the success ratio of overhearing in Terminal n for pieces addressed to Terminal m, and is updated every time the server receives the piece from the terminal in Algorithm 4. Before calculation of the smoothing ($0 \le \beta \le 1$), $Terminals[n].R[m]$ is set as 1 for the case where the piece is received, and 0 for the case where the piece is not received.

3.3 Method C: Network Coding for Pieces

Overhearing is an effective method for piece distribution when terminals request the same pieces. However, overhearing in Method B does not work effectively when almost all terminals are ending the download because each terminal requests different pieces. In Method C, the server uses network coding to transmit multiple pieces as one coded piece in the application layer in the case where each terminal needs different pieces. Then, the receivers can obtain the desired piece by decoding the coded piece via overhearing.

The server selects the destination and the pieces to transmit with Method B. If the number of expected receivers (Eq. (4)) is increased by network coding, network coding is applied. When the coded piece is decoded in the receivers to obtain the desired pieces, the receivers have to have piece(s) except for the desired piece. The server calculates the number of expected receivers for every combination of pieces which meets the condition for decoding, and selects the combination with the highest number at

Algorithm 5 Piece selection for Network Coding.

▷ Destination terminal n and Piece m are selected.
$terminalList \leftarrow$ [ ] ▷ Piece m, to Expected receiver.
for $i = 0$ to $Terminals$.length−1 **do**
  if $Terminals[i].Pieces[m]$ = 'requested' **then**
    $terminalList$.add($Terminals[i]$)
  end if
end for
$pids \leftarrow$ [ ] ▷ Arrary of Pieces for Network coding.
loop
  ▷ Make Array of Pieces for all in $terminalList$.
  $intersectList \leftarrow terminalList$[0].indexesOf('have')
  for $i = 1$ to $list$.length−1 **do**
    $intersectList \leftarrow$
      $intersectList$.intersect(
        $terminalList[i]$.indexesOf('have'))
  end for
  if $intersectList$.length = 0 **then**
    ▷ End if Receiver has no piece.
    Break
  end if
  $score_{\max} \leftarrow 0$
  $pid_{\max} \leftarrow null$
  $dests_{\max} \leftarrow$ [ ]
  $terms \leftarrow Terminals - terminalList$
  ▷ # of expected terminals, select Max.
  for $pid = 0$ to $intersectList$.length−1 **do**
    $score \leftarrow 0$
    $dests \leftarrow$ [ ]
    for $i = 0$ to $terms$.length−1 **do**
      ▷ Terminals requesting Piece i will decode.
      if $terms[i].Pieces[pid]$ = 'requested' **then**
        $score \leftarrow score + terms[i].R[n]$
        $dests$.add($terms[i]$)
      end if
    end for
    if $score > score_{\max}$ **then**
      $score_{\max} \leftarrow score$
      $pid_{\max} \leftarrow pid$
      $dests_{\max} \leftarrow dests$
    end if
  end for
  ▷ End if no piece for Network coding.
  if $dests_{\max}$.length = 0 **then**
    Break
  end if
  ▷ Add Receiver expecting.
  $terminalList \leftarrow terminalList + dests_{\max}$
  $pids$.add($pid_{\max}$)
  ▷ End if # of Receivers == # of all terminals.
  if $terminalList$.length = $Terminals$.length **then**
    Break
  end if
end loop
▷ Return an Array of Piece ID for Network coding.
return $pids$

that time.

The procedures used in Method C are described in Algorithm 5. In the algorithm, the destination terminal n and the piece m are selected for transmission. First, Algorithm 5 lists commonly having pieces (*intersectList*) of terminals requesting piece m. These listed pieces can be decoded by these terminals if the pieces are coded with piece m for transfer. If no common piece exists in terminals requesting piece m, network coding is not performed. Then, for all cases of listed piece(s), the possibility of decoding at the terminals is calculated when these pieces are coded with piece m, and the combination with the largest number of pieces is selected. By repeating these procedures while there are possible combinations for the network coding the server selects the pieces. Note that a method of 'indexesOf' returns the same array index with the parameter, a method of 'intersect' returns the same array element with the array parameter, and an 'add' method adds the parameter to an array.

4. Evaluation

4.1 Evaluations in Testbed

4.1.1 Evaluation Setup

The proposed method was implemented in a user space application running on Linux, and laptop PCs with a wireless interface were used as the hardware. The application established a TCP connection with the content distribution server, requested the pieces, and received them. At the same time, the application opened the raw socket in promiscuous mode to process the overheard frames.

We aim to evaluate the basic performance of the proposed method in the case that the terminals start downloading at the same timing, and at different timings. The content size of documents, videos or photos is supposed, and the size is enough to evaluate the scheduling of pieces and terminals. In the experiments, we prepared one server (Terminal 1) and 6 receivers (Terminals 2–7). Terminals 2 and 3 are placed near the distribution server (1–2 m), and Terminal 4–7 are placed further away from the server (10–12 m) in our laboratory. Then, all the terminals join the group to download content because the proposed method does not suppose to join a terminal whose the transfer speed of MAC from the server is very low. We evaluated Methods A, B, and C in the situation where all of them started downloading at the same time (Simultaneous start), and Terminals 2–7 started downloading from the server at 5-second intervals (Sequential start). In case of Sequential start, the downloads start in order from Terminal 2 to 7. **Tables 1** and **2** show the experiment environment and conditions. α in Algorithm 3 and β in Algorithm 4 are 0.8 and 0.8, respectively.

4.1.2 Results

The average throughput is shown in **Table 3**, and the average throughput for each terminal is shown in **Fig. 4** and **Fig. 8**. The throughput for Method C is 4.3 times faster than that of HTTP for a simultaneous start, and is 3.2 times faster than that of HTTP for a sequential start. The piece and the destination selection method improve the performance when we compare Methods A–C. With regard to the average throughput of each terminal, the improvement of Terminal 7, which had the lowest throughput, was the greatest as a result of changing the selection method for both start times. In Method B, the throughput of some of the terminals is reduced because a reduced bandwidth is used in order to improve the throughput of terminals having a low throughput. However, the network coding employed in the case of Method C improved the performance of the terminal with a low throughput without reducing the throughput of the other terminals. However, the HTTP server transmits content with 20.64 Mbps (6 terminals × 3.44 Mbps) but the average throughputs of Method A–C are smaller than the server throughput. This is because the server tends to select terminals with a low transmission speed of MAC to extend the overhearing range, and then this control reduces the efficient use of the bandwidth of MAC. We discuss this trade-off in Section 5. Note that the average throughput of HTTP is almost the same even though the transmission speed of the physical layer for each terminal is different in Fig. 4. Since CSMA/CA fairly shares opportunities of frame transfer for each terminal, the shared bandwidth for each terminal is almost the same if frame losses do not affect the congestion control of TCP [14]. Evaluations in this paper assume wireless conditions are not extremely worse and the retransmission controls of IEEE 802.11a recovers frame losses.

Table 1 Environment of the experiment.

Server, Terminal	DELL vostro 3550 (Core i3 2.10 GHz, Mem: 3.5 GB)
OS	Debian GNU/Linux 6.0 (Kernel 2.6.32-5)
WLAN interface	Buffalo WLI-EXC-AG300N
WLAN standard	IEEE802.11a
WLAN channel	5220 MHz (44 ch)
WLAN mode	Ad-hoc, ARF ON

Table 2 Conditions of Experiment.

Size of distribution file	100 MB
Size of piece	128 KB
Number of terminals	6
Interval of piece having message	0.5 sec
Overhearing timeout	1.0 sec

Table 3 Average throughput.

Method	Simultaneous Start	Sequential Start
HTTP	3.44 Mbps	4.13 Mbps
Method A	10.3 Mbps	10.9 Mbps
Method B	11.0 Mbps	12.5 Mbps
Method C	14.8 Mbps	13.4 Mbps

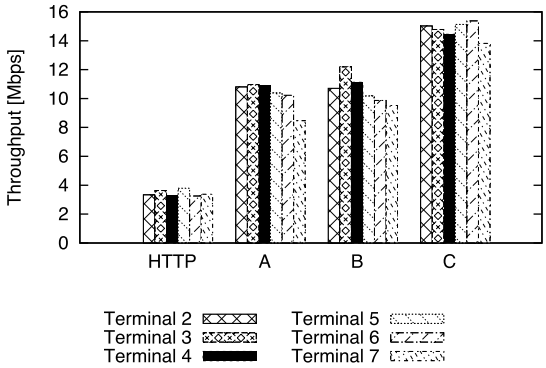


Fig. 4 Throughput of HTTP, and Method A-C (Simultaneous start).

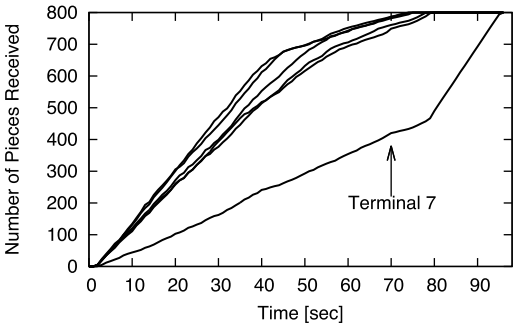


Fig. 5 Cumulative number of pieces received for each terminal (Method A, Simultaneous start).

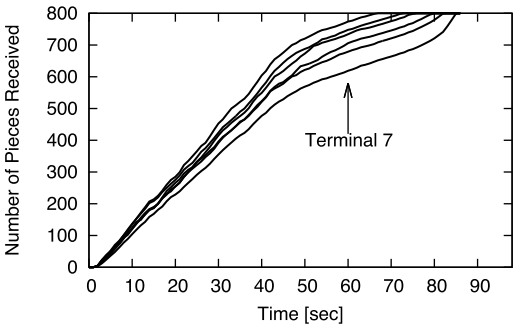


Fig. 6 Cumulative number of pieces received for each terminal (Method B, Simultaneous start).

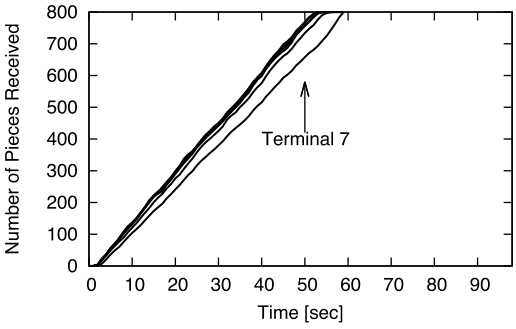


Fig. 7 Cumulative number of pieces received for each terminal (Method C, Simultaneous start).

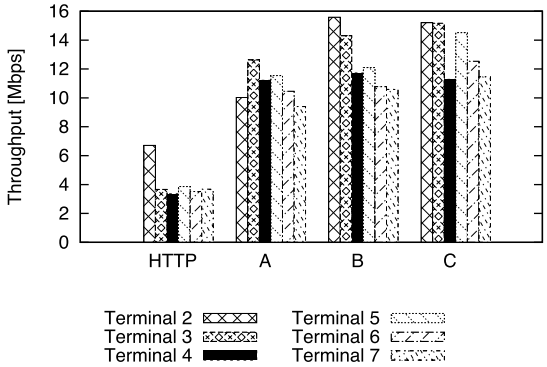


Fig. 8 Throughput of HTTP, and Method A-C (Sequential start).

Figures 5–7 and **9–11** show the cumulative number of pieces received by each terminal for Methods A–C. In the case of a simultaneous start (Figs. 5–7), the throughput of terminal 7 had the lowest throughput in Method A because the terminal 7 cannot overhear frames of the other terminals. Since the terminal 7 is often selected as the destination in Method B, the throughput of the terminal 7 is improved by 12%. In Methods A and B,

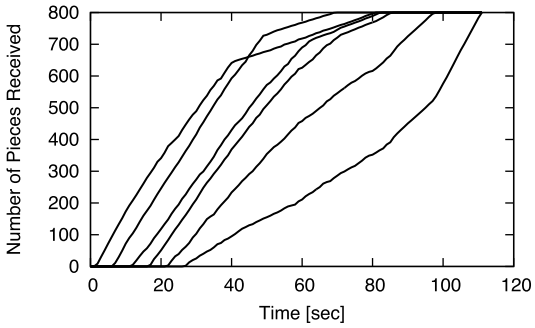


Fig. 9 Cumulative number of pieces received for each terminal (Method A, Sequential start).

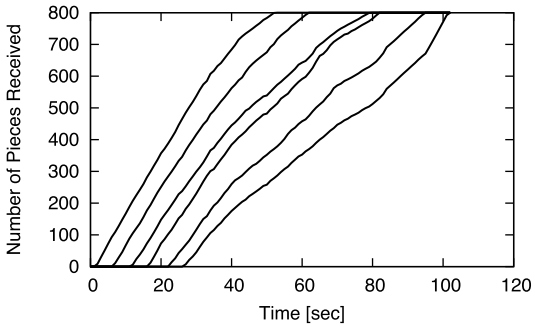


Fig. 10 Cumulative number of pieces received for each terminal (Method B, Sequential start).

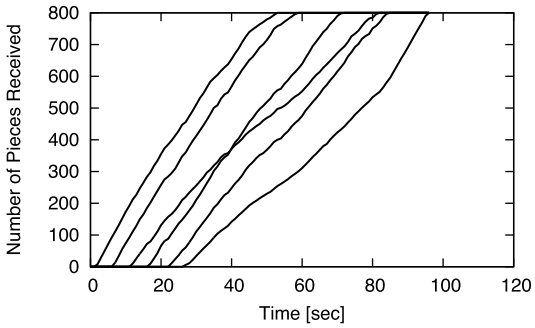


Fig. 11 Cumulative number of pieces received for each terminal (Method C, Sequential start).

the throughputs are reduced after 40 seconds, but Method C improved the reduction of throughput and the average throughput by 35%. Since each terminal requests different pieces at the end of downloading where the start was simultaneous, network coding has a significant effect.

In the case of a sequential start (Figs. 9–11), the throughput reduction at the end of downloading in Method A is improved by 17% in Method B. In this case, the improvement in throughput is larger compared to that obtained with a simultaneous start. Since the download status is different for each terminal, a terminal at the end of downloading is often selected as the destination by the controls of Method B. For Method C, the average throughput is improved by 7% compared to Method B. This is because there are fewer cases where each terminal requests different pieces compared to a simultaneous start. There is therefore a lower improvement in throughput compared to a simultaneous start. Note that the performance of terminal 4 is low for Method A—C because terminal 4 in Fig. 8 cannot overhear the other terminals due to its wireless conditions. Then, since Method C works to improve ef-

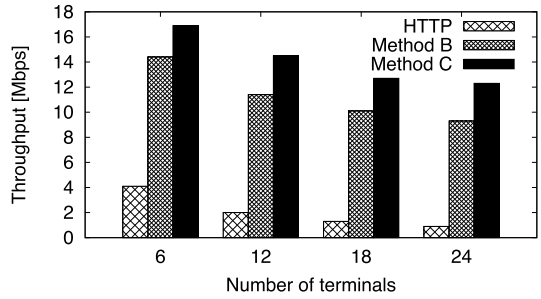


Fig. 12 Average throughput for the number of terminals (Simultaneous start).

ficiency by decoding the network coded pieces via overhearing, the network coding does not effectively work for terminal 4.

4.2 Evaluations in the Emulation Environment

Next, we evaluate the scheduling A–C in case that the number of terminals is increased in the emulation environment. As a distributed emulation environment, we use EMANE [13]. However, since the original EMANE has problems for dealing with overhearing frames in MAC layer, we fixed the problems [15]. Parameters setting of the emulation environment is to get almost the same results of a testbed. We confirmed by checking it with the overhearing conditions and throughput in 6 terminals case. In the following evaluations, the same setting is used except for the number of terminals. Since the terminals are placed at the same distance from the server terminal, results of Method A are omitted in this section.

Figure 12 shows the average throughput of one terminal while changing the number of terminals in case the terminals start downloading simultaneously. For increasing the number of terminals, the average throughput for the HTTP case is reduced from 4.1 to 0.9 Mbps. Since overhead of MAC and the proposed method are increased for increasing the number of terminals, the average throughput of the proposed methods are also reduced. The average throughput of Method B and Method C are also reduced from 14.4 to 9.3 Mbps and from 16.9 to 12.3 Mbps, respectively. However, these methods keep 10–13 times throughput for that of HTTP in case of 24 terminals. In Method C, the reduction is smaller than that of Method B because the network coding can reduce the number of transmissions by transmitting multiple pieces as one XORed piece. With using the network coding, the proposed method works effectively even if the number of terminals are increased.

5. Discussion

In the proposed method, the server terminal has to run Algorithms in Section 3 to select pieces and terminals. In case the number of terminals or pieces are increased, the calculation complexity is also increased and it will cause a heavy load on the server terminal. For Method B and C, although the calculation complexity is suppressed by adding a restriction for possible destination terminals and pieces to send, the complexity is still large. We confirmed the proposed method can work for a group of a few dozen terminals in case of conditions of Section 4.1. However, it will be difficult to accommodate over one hundred terminals of a group. We need more sophisticated algorithms to reduce the calculation complexity without reducing the accuracy of the selections for the scalability issue.

As we mentioned in Section 4.1.1, the proposed method does not suppose to join a terminal whose the transfer speed of MAC from the server is very low. Then, the server terminal controls to be the range of overhearing large with selecting a terminal with the low transfer speed terminal as the destination. However, this control may reduce the download speed of terminals in the group because the transmission speed for the low wireless link quality terminal is slow. Then, we have to carefully choose terminals to join the group with considering wireless link quality because there is a tradeoff between a range of overhearing and a transmission speed of MAC. In case that a terminal with low wireless link quality joins the group, the server may have to exclude the terminal from the group not to reduce the download speed of terminals in the group with considering the tradeoff. It will be the future work of our study.

6. Conclusion

In this paper, we have proposed a piece and a detonation terminal selection method that enables a reliable content distribution in a single hop wireless ad hoc network by adding file piecing to the pseudo multicast scheme. The scheduling takes into account in the case where communication conditions are different for each terminal, and the download status of each terminal. We also use network coding to improve the overhearing efficiency. We developed a testbed, and the performance of our method was evaluated on the testbed. The results of the evaluation show that network coding improves the download speed regardless of whether downloading of content begins at the same or different times. In future research, we apply the method to multi hops wireless communication to deal the terminals in a poor wireless communication condition, and to accommodate a large number of terminals.

Acknowledgments This work is supported by KAKENHI (23700071).

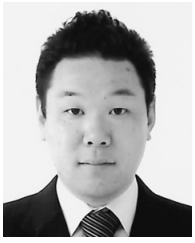
References

[1] Lin, J.C. and Paul, S.: RMTP: A reliable multicast transport protocol, *Proc. IEEE INFOCOM'96*, Vol.3, pp.1414–1424 (1996).
[2] Holland, O. and Aghvami, H.: Dynamic Switching between One-to-Many Download Methods in "All-IP" Cellular Networks, *IEEE Trans. Mobile Computing*, Vol.5, No.3, pp.274–287 (2006).
[3] Dujovne, D. and Turletti, T.: Multicast in 802.11 WLANs: Experimental study, *Proc. 9th ACM MSWiM*, pp.130–138 (2006).
[4] Chandra, R., Karanth, S., Moscibroda, T., Navda, V., Padhye, J., Ramjee, R. and Ravindranath, L.: DirCast: A practical and efficient Wi-Fi multicast system, *Proc. IEEE ICNP*, pp.161–170 (2009).
[5] Dogar, F.R., Phanishayee, A., Pucha, H., Ruwase, O. and Andersen, D.G.: Ditto: A system for opportunistic caching in multi-hop wireless networks, *Proc. Mobicom*, pp.279–290 (2008).
[6] Shen, S.-H., Gember, A., Anand, A. and Akella, A.: REfactor-ing content overhearing to improve wireless performance, *Proc. MobiCom*, pp.217–228 (2011).
[7] The BitTorrent Protocol Specification, available from ⟨http://www.bittorrent.org/beps/bep_0003.html⟩.
[8] Krifa, A., Sbai, M.K., Barakat, C. and Turletti, T.: BitHoc: A content sharing application for Wirless Ad hoc networks, *Proc. IEEE PerCom*, pp.1–3 (2009).
[9] Katti, S., Rahul, H., Wenjun, Hu, Katabi, D., Medard, M. and Crowcroft, J.: XORs in the Air: Practical Wireless Network Coding, *IEEE Trans. Networking*, Vol.16, No.3, pp.497–510 (2008).
[10] Rozner, E., Iyer, A.P., Mehta, Y., Qiu, L. and Jafry, M.: ER: Efficient

retransmission scheme for wireless LANs, *Proc. ACM CoNEXT Conference*, pp.8:1–8:12 (2007).
[11] Nguyen, D., Tran, T., Nguyen, T. and Bose, B.: Wireless Broadcast Using Network Coding, *IEEE Trans. Vehicular Technology*, Vol.58, No.2, pp.914–925 (2009).
[12] Yokose, H., Ohzahata, S. and Kato, T.: An Efficient Contents Distribution Method Using Overhearing and File Piecing in Wireless LAN, *Proc. APSITT*, pp.1–6 (2012).
[13] Natalie, I., Rivera, B. and Adamsonk, B.: Mobile ad hoc network emulation environment, *Proc. IEEE MILCOM*, pp.1–6 (2009).
[14] Heusse, M., Rousseau, F., Berger-Sabbatel, G. and Duda, A.: Performance anomaly of 802.11b, *Proc. INFOCOM*, pp.836–843 (2003).
[15] Nitta, K., Ohzahata, S. and Kato, T.: Improving Cause-and-effect Relationships among Single-hop Neighbors of Wireless Ad-hoc Network for Parallel Network Emulation, *IEICE Trans. Commun.*, JB-98, No.2, pp.107–115 (2015) (in Japanese).

Hiroaki Yokose received his M.E. degree from the University of Electro-Communications, Tokyo, Japan, in 2013. His research interests include content distribution for wireless environments.

Koji Nitta received his M.E. degree from the University of Electro-Communications, Tokyo, Japan, in 2014. His research interests include distributed emulation environment for wireless network.



Satoshi Ohzahata received his B.S., M.E., and D.E. degrees from University of Tsukuba in 1998, 2000 and 2003, respectively. He was a Research Associate, Department of Computer, Information & Communication Sciences at Tokyo University Agriculture and Technology from 2003–2007, and was an assistant professor of the same university from 2007–2009. Since 2009, he has been an associate professor at Graduate School of Information Systems, the University of Electro Communication. His interests are mobile ad hoc networks, the Internet architecture in mobile environments and Internet traffic measurement. He is a member of IPSJ, IEEE, ACM and IEICE.

Toshihiko Kato received his B.E., M.E. and Dr. Eng. degrees electrical engineering from the University of Tokyo, in 1978, 1980 and 1983, respectively. He joined KDD in 1983 and worked in the field of communication protocols of OSI and Internet until 2002. From 1987 to 1988, he was a visiting scientist at Carnegie Mellon University. He is now a professor of the Graduate School of Information Systems in the University of Electro-Communications in Tokyo, Japan. His current research interests include protocol for mobile Internet, high speed Internet and ad hoc network.