Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9202151, 9 pages
http://dx.doi.org/10.1155/2016/9202151

Research Article

Group-Interest-Based Verifiable CCN

DaeYoub Kim

Department of Information Security, Suwon University, 17 Wauan-gil, Bongdam-eup, Haseong-si, Gyeonggi-do 445-743, Republic of Korea

Correspondence should be addressed to DaeYoub Kim; daeyoub69@suwon.ac.kr

Received 11 September 2015; Accepted 29 December 2015

Academic Editor: Kamal Deep Singh

To solve various problems of the Internet, content centric networking (CCN), one of information centric networking architectures (ICN), provides both an in-network content caching scheme and a built-in content verification scheme. However, a user is still asked to generate many request messages when retrieving fragmented content through CCN. This model can seriously increase the amount of network traffic. Furthermore, when receiving content, a user is asked to verify the received content before using it. This verification process can cause a serious service delay. To improve such inefficiencies, this paper proposes a transmission process to handle request messages at one time. Also, it suggests an efficient content verification method using both hash chains and Merkel-hash tree.

1. Introduction

The Internet was originally designed to establish reliable connections between remotely located hosts [1]. The initial designers of the Internet did not expect that the Internet would be utilized for various services/applications as now. Also, they did not consider various problems which are currently faced by the Internet: as various services/applications begin to utilize the Internet, the amount of network traffic rapidly increases, which leads to serious network congestion [2]. For example, as mobile/smart consumer devices are popularized, it becomes trendy for users to actively generate/share content from their daily lives with others using their own mobile/smart devices. Also, the quality of shared content has become higher than that of the past [3]. Furthermore, various IoT (Internet of Things) services like a vehicle communication system gather/provide massive amounts of information through the Internet [4].

Besides a network congestion problem, the weak security of the Internet is also a serious problem which should be improved [5, 6].

To solve such problems of the Internet, various future Internet architectures/technologies like information centric networking architecture (ICN) are introduced [7]. Specially, since ICN is focusing on contents itself, not on a host providing the content, ICN can make it possible that a user receives content from several possible hosts caching the content. So a user can access content through ICN more efficiently as well as more rapidly than through the Internet [8–10].

Content centric networking architecture (CCN) is one of ICN [11, 12]. CCN has several distinguishing characteristics as follows:

(i) It is designed as a request-driven communication model.

(ii) It utilizes in-network caching functionality to enhance network efficiency.

(iii) It delivers network packets referring to a content identity, not to a device identity (e.g., IP/MAC address) so as to efficiently use cached content.

(iv) It provides a built-in content verification mechanism to authenticate both received content and the original publisher of the content.

However, such characteristics of CCN still cause network/computation inefficiencies. Actually, to distribute content through CCN, the content is fragmented into several segments with small size, and each segment of the content is handled as an independent data in CCN. Hence, when requesting the content, a user should generate a request

message for each segment of the content. These request messages can increase the amount of network traffic so as to be misused by denial-of-service (DoS) attackers [13, 14].

Furthermore, since CCN utilizes content cached in intermediated nodes, it is possible that a user receives content from unknown (malicious) nodes, not from the original publisher of the content. So CCN highly recommends that a user verifies received content before using the content. This content verification process could solve certain security problems of the Internet such as malware and man-in-the-middle attacks [15]. However, since a user is asked to verify all segments of content, such a recursive verification process can cause long service delays [16, 17].

Hence, utilizing CCN for various IT services like multimedia content distribution services as well as various IoT services, the transmission and computation overheads of CCN should be improved [18]. Hence, this paper proposes a process to handle a set of serial request messages at one time to enhance the network efficiency of CCN as well as an improved content verification mechanism to reduce the service latency of CCN.

2. Content Centric Networking

To enhance network efficiency, CCN implements a content-caching functionality on network nodes. Then if a node caching content receives a request packet (Interest) for the cached content, the node transmits the cached content as a response packet (data) to the sender of the Interest and then finishes forwarding the Interest. Hence, a user can receive the content more rapidly than when receiving the content from the original provider of the content. Also, since request messages that converged to the original provider of the content can be handled by intermediated nodes, CCN can solve a network congestion problem which can happen close to the content provider.

Also, to efficiently use the cached content, CCN utilizes the hierarchical identity of content as a packet forwarding address. Since this hierarchical identity of content should be uniquely defined in network, when receiving Interest, an intermediated node can search cached content in its storage (content store, CS) just analyzing the forwarding address of the Interest. The hierarchical identity of content is called a content name.

Figure 1 describes CCN process to handle Interest/data:

(1) If a user generates/sends Interest for a segment of content (e.g., a.mpg), an intermediated node receives the Interest through its interface (e.g., Face 1).

(2) The node checks whether the requested segment has been cached in CS. If it has been, the node sends back the cached segment through Face 1 as data. Then the node completes the processing of the received Interest.

(3) If the requested segment is not cached in CS, the node checks its pending Interest table (PIT) to confirm whether it has already forwarded the same Interest. If the node did, since the content name of the Interest has been recorded in its PIT, the node can find an entry of PIT which is relevant to the Interest. In this case, the node just adds Face 1 on the found entry of PIT, and then stops handling the Interest.

(4) If there is no found entry of PIT, the node compares the content name of the Interest with the entries of its forwarding information based (FIB) table using the longest prefix match in order to select a proper interface (e.g., Face 3) through which it will forward the Interest.

(5) The node records both the content name of the Interest and the incoming interface (Face 1) of the Interest on its PIT.

(6) The node forwards the Interest through Face 3.

(7-8) When receiving data, the node checks whether there is an entry of PIT which is matched to the content name of the data. If there is no proper entry of PIT, the node discards the data and then stops handling the data.

(9-10) If there is a proper entry of PIT, the node saves the data in CS and then forwards the data through the faces of the found entry of PIT. Specially, if the node is an end-user's device, it should first check the validity of the data and then save the data in CS only if the data is valid. Finally, the node deletes the found entry of PIT.

Table 1: The structure of a (Group-) Interest.

Interest structure		
BYTE []	name;	
INT	*n*Seg;	
INT	*g*Seg;	// optional
INT	*t*Seg;	// optional
INT	version;	

3. Group-Interest Operation

As shown in Figure 1, to transmit content, it is first required to generate/send Interest. Specially, CCN asks a content publisher to fragment content into several segments with small size to distribute the content. Then CCN deals with each segment of the content as a single data. So, for receiving the content, a user should generate/send many Interests, even though the only difference of these Interests is just the number of segment. This requesting process may increase the amount of network traffic. Also, after receiving the $i-1$th segment of content, a user can generate/transmit Interest for the ith segment of the content. This linear process can lead to long content retrieval latency.

To improve such problems, we suggest a Group-Interest for requesting m serial segments at one time. Table 1 shows an Interest structure for a Group-Interest:

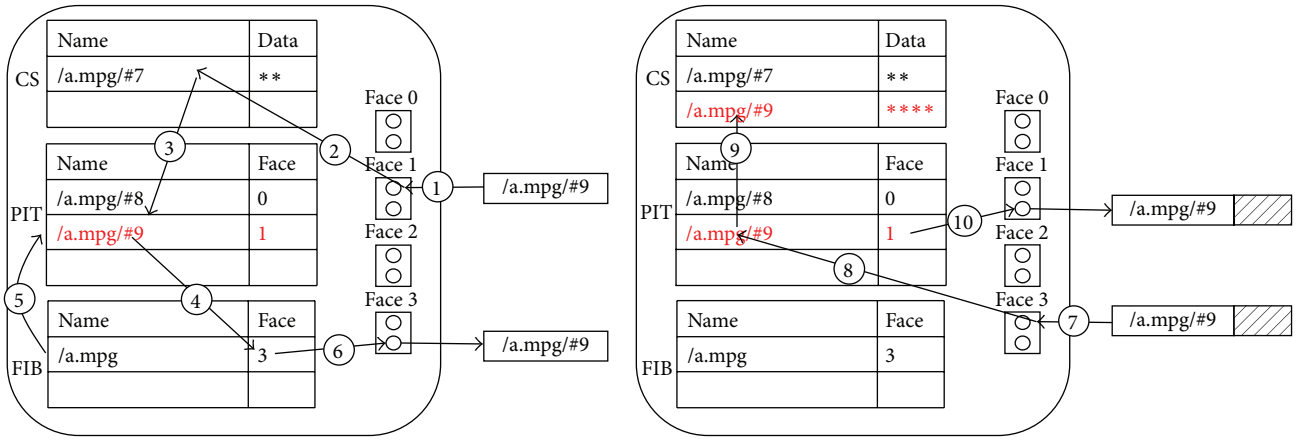
(i) [name] is the hierarchical prefix identities of content.

(ii) [*n*Seg] is a serial number of the segment of the content.

(iii) [version] is the publication time of the content.

Actually, these three fields are the original fields of Interest. The following two optional fields are added for a Group-Interest:

(i) [*g*Seg] describes the number of segments which this Group-Interest requests. That is, this Group-Interest



(a) Interest process (b) Data process

Figure 1: CCN forwarding model using Interest and data.

Table 2: The structure of PIT to handle a (Group-) Interest.

PIT_Entry structure		
BYTE []	name;	
INT	num;	//segment num.
INT	version;	
INT []	face;	
INT	time;	//expire time
INT	gFlag	//group flag
INT	sFlag	//group size
BOOL	rFlag	//response flag

requests serial segments with identities from nSeg to nSeg + gSeg − 1. If gSeg = 1, this Interest is a general Interest, not a Group-Interest.

(ii) [tSeg] is the total number of segments which consists of the requested content. This field is used for verifying gSeg. That is, if nSeg + gSeg − 1 > tSeg, gSeg is invalid.

To handle a Group-Interest, it is necessary to modify the structure of PIT entry as shown in Table 2: [gFlag], [sFlag], and [rFlag] are added.

(i) [gFalg] describes whether this PIT entry is relevant to a Group-Interest or not. If gFlag = 0, this PIT entry is for a general Interest. In this case, both sFlag and rFlag are unmeaning. Otherwise, it means that this PIT entry is relevant to a Group-Interest. Specifically, the value of gFlag is the number of the first segment of the Group-Interest.

(ii) [sFlag] is the number of segments which the Group-Interest requires. This field can be used to delete entry of PIT.

(iii) [rFlag] describes whether the relevant data has been received or not. If rFlag = 1, it means that the data has been transmitted and then forwarded toward requesters.

```
Interest Operation Code
Input: Interest, Face
Output: void
delete expired entries of PIT; // call DeleteEntryOfPIT( );
set f-Flag = 0; // forwardingFlag
for each index i from nSeg to nSeg + gSeg −1 {
  generate Interest[i] such that
    Interest[i].name == Interest.name and
    Interest[i].nSeg == Interest.nSeg;
  find an entry (E[i]) of PIT relevant to Interest[i];
  if there is no, add a new entry to its PIT for Interest[i];
  else {
    if Face isn't in E[i].face, add Face to E[i].face;
    if E[i].gFlag > 0 and E[i].rFlag == 1, set E[i].rFlag = 0;
    else stop handling Interest[i];
  }
  find an entry (C[i]) of CS relevant to Interest[i]
  if there is C[i], then {
    transmit C[i] through E[i].face;
    if E[i].gFlag == 1, then set E[i].rFlag = 1.
    else delete E[i] from PIT; // general Interest
  }
  else set f-Flag = 1;
  if E[i].gFlag > 0 and E[i].rFlag == 0, set f-Flag = 1;
}
find a proper forwarding face referring to FIB table;
if gSeg == 1, forward Interest through the face;
else if f-Flag == 1, forward Interest via the face;
else stop handling Interest;
```

Pseudocode 1: The pseudocode to handle a Group-Interest.

Pseudocodes 1 and 2 are pseudocodes describing how to handle both a general Interest and a Group-Interest. As described in Pseudocode 1, the major differences between the

```
Delete Entry Of PIT Code
Input: void
output: void
for each index i from 1 to sizeOfPIT {
   read the ith entry (E[i]) of PIT;
   if E[i] is expired, delete E[i];
   if E[i].gFlag > 0 and E[i].rFlag == 1, {
      set deleteFlag = 1;
      for each k from 1 to sizeOfPIT {
         if E[k].name == E[i].name and E[k].gFlag ==
         E[i].gFlag and E[k].rFlag == 0, then deleteFlag = 0;
      }
      if deleteFlag == 1, delete E[i];
   }
}
```

Pseudocode 2: The pseudocode to handle PIT.

processes of a general Interest and of a Group-Interest are as follows:

(i) A Group-Interest is disassembled to generate general Interests. These Interests are, respectively, corresponding to serial segments requested by the Group-Interest. These disassembled Interests are only internally used for managing PIT. That is, when handling PIT, a node uses these disassembled Interests, not the original Group-Interest.

(ii) Each entry of PIT generated from a Group-Interest is deleted when either the entry has expired or after all segments requested by the Group-Interest have been forwarded to requestors.

(iii) A Group-Interest is forwarded until all segments requested by the Group-Interest have been transmitted to requestors.

Since the proposed process of Interest as shown in Pseudocode 1 can handle a Group-Interest as well as a general Interest, a user can selectively generate either a general Interest or a Group-Interest considering response status. That is, after receiving some data packets relevant to a Group-Interest, to request remaining data again, a user can selectively generate either a general Interest or a Group-Interest.

Also, it is necessary to modify the process of data in order to handle a Group-Interest. Specially, it is needed to prevent duplicated packet transmission. For that, Pseudocode 3 shows a modified process. The major differences of the modified data process are as follows:

(i) If data is relevant to a Group-Interest and the same data has been forwarded already, a node does not forward the data again even though the relevant entry of PIT exists.

(ii) If data is relevant to a Group-Interest, relevant PIT entry is not instantly deleted from PIT.

```
Data Operation Code
Input: Data
output: void
find an entry (E) of PIT corresponding to Data;
if there is no, stop this process;
else {
   if E.gFlag and E.rFlag are all 1, stop this process;
   save Data in CS;
   forward Data through E.face;
   if E.gFlag is 0, delete E from PIT;
   else set E.rFlag = 1;
}
```

Pseudocode 3: The pseudocode to handle data.

4. Content Verification

In CCN, since a node can receive a segment of content from an anonymous network node caching the segment as well as from the original publisher of the segment, it is possible that malicious nodes send a forged segment.

Hence, a content verification process is one of the essential requirements of CCN. However, since a user should recursively verify each segment of content whenever the user receives the segment, this recursive verification process can cause another inefficiency of CCN.

4.1. MHT-Based Content Verification Scheme. To efficiently verify both the segments of content as well as the original publisher of the content, CCN utilizes a Merkel-hash tree (MHT) [11, 19–21]. Figure 2 shows an example of a MHT-based content verification scheme: assume that a content-publisher fragments content into 7 segments $\{S_2, \ldots, S_8\}$ and then generates meta-data S_1 describing the structure of the segments of the content. From now on, we assume that content consists of 8 segments including a metadata segment.

Step 1 (constructing MHT). A content-publisher builds a binary tree with 8 leaf nodes and then assigns $\{S_1, \ldots, S_7, S_8\}$ to leaf nodes in numerical order. Then the publisher computes the hash value $H(S_i)$ of each segment S_i using the one-way hash function H. The publisher uses $H(S_i)$ as the node value V_k of a leaf node N_k which is assigned to S_i.

Step 2 (computing node values). For each node N_j except for leaf nodes, the publisher computes a node value $V_j = H(V_{2j} \parallel V_{2j+1})$, where $\parallel$ is a concatenation operation and N_j is the parent node of two child nodes, N_{2j} and N_{2j+1}.

Step 3 (signing a root node value). After computing all node values of the binary tree, the publisher signs a root node value V_1 with its signature key SK to generate a signature value (sign).

Step 4 (generating a witness of a segment S_i). For each segment S_i, let N_ks be the sibling nodes of the nodes on the path, from a leaf node assigned to S_i to the root node N_1. The

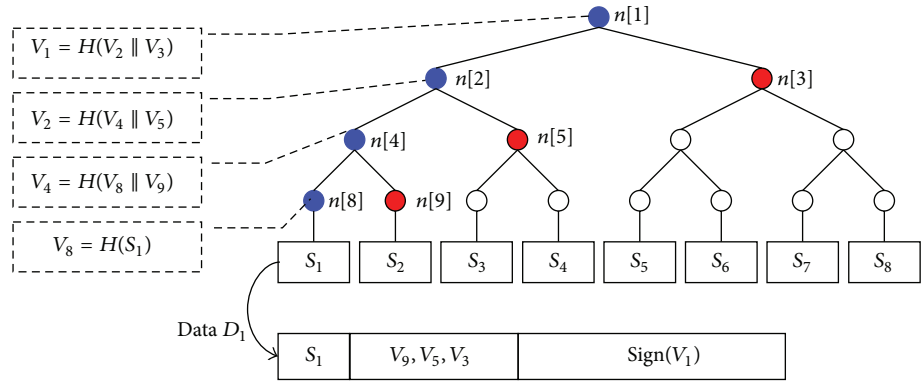
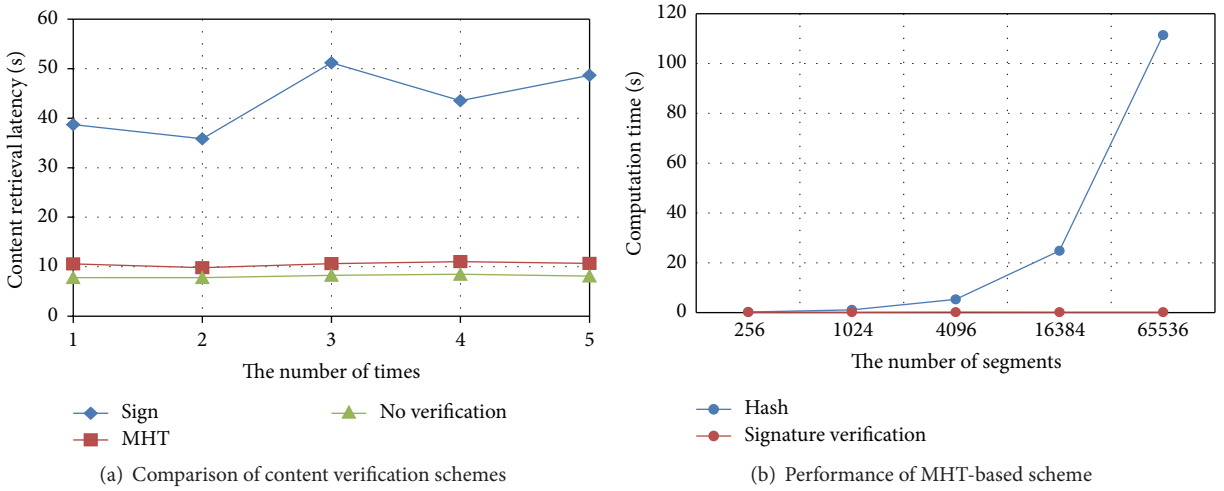


FIGURE 2: MHT-based contents verification: each CCN data contains a segment, a relevant witness, and a signature.



(a) Comparison of content verification schemes

(b) Performance of MHT-based scheme

FIGURE 3: Operation delay. (a) It describes the comparison of response time to share contents fragmented into 256 segments between two smart phones. (b) In the case of applying MHT-based scheme for content.

publisher generates a witness W_i which consists of the node values, V_ks, of N_ks. For example, the W_1 of S_1 is $\{V_9, V_5, V_3\}$ in Figure 2. The witness is needed to verify the sign. That is, using both S_i and W_i, any verifier can compute the same V_1 and then verify the sign.

Step 5 (packaging as data). The publisher generates data (D_i) packaging S_i, W_i, and sign.

If a user receives D_i, the user recursively computes the necessary hash values using both S_i and W_i to compute the root node value, V_1. Then the user verifies the packaged sign using the computed V_1. In practice, after verifying the sign packaged in the first type of data, D_1, the user temporarily saves the computed V_1. Then, the user does not need to verify the sign again for verifying S_i ($i > 1$). Instead, it is sufficient that the user just compares the computed V_1 with the previously saved V_1. Hence, it is possible to reduce the operation time of a segment verification process.

However, as shown in Figure 3, the operation delay of a MHT-based verification scheme is still a burden to CCN. Figure 3(a) shows the comparison result of response times when sharing 256 segments of content between two smart phones over WLAN using three different methods:

(i) [No verification] is a case that a user does not verify received data at all.

(ii) [Sign] is a case that each data has a relevant signature value in order that any user receiving the data can instantly verify the data.

(iii) [MHT] is a case to verify data using MHT.

The result shows that a MHT-based verification scheme can reduce the response time needed to verify segments as compared with [Sign]. That is, [MHT] is more efficient than [Sign] by about 75%.

However, as shown in Figure 3(b), the computation overhead of MHT has increased proportionally to the number of segments, that is, to the size of content. It means that a MHT-based verification process can still cause a serious service delay when distributing high-quality, large-size content. This overhead is due to the fact that the number of recursive hash operations of a MHT-based scheme increases. Hence, to improve the performance of the content verification process

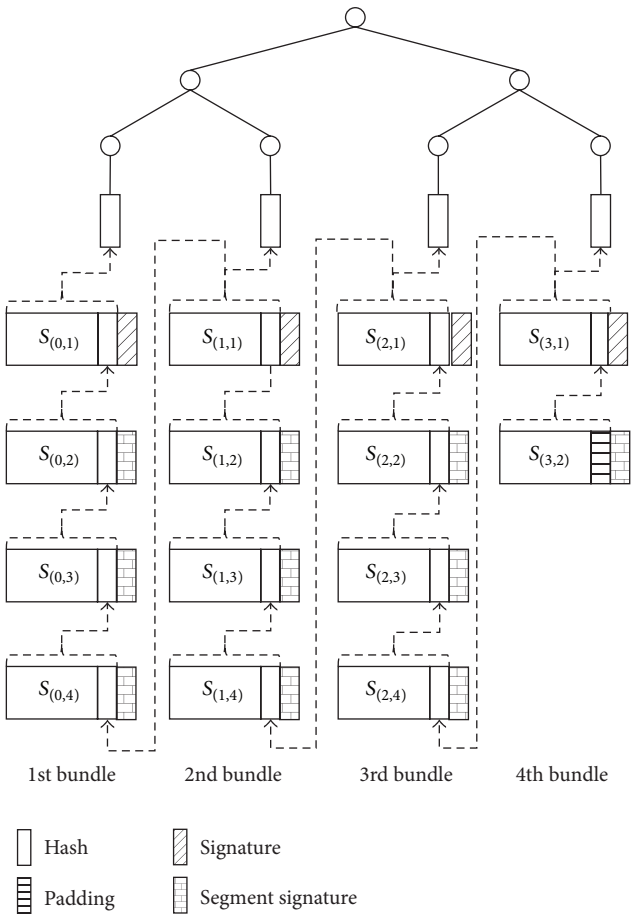


Figure 4: Bundle Interest-based CCN content verification using H-MHT.

of CCN, it is necessary to reduce the total number of hash operations needed to verify the segments of content.

4.2. Hash Chain Based MHT Verification. To reduce an operation delay caused by the content verification process of CCN, in this section, it is proposed to combine MHT with a hash chain which is a general approach to reduce the amount of verification data. It is called a hash chain based MHT scheme (H-MHT).

4.2.1. Verifiable Data Generation. As shown in Figure 4, H-MHT utilizes both MHT and hash value chains: let the number ($N = 2^n$) of the leaf nodes of MHT be 4. Let the number (S) of the segments of content be 14. That is, the content consists of 14 segments, $\{S_1, \ldots, S_{14}\}$ including a metadata segment.

Step 1 (bundling segments). The content-publisher divides 14 ($= S$) segments into 4 ($= N$) segment bundles $\{B^0, B^1, B^2, B^3\}$. Let b_i be the size of a bundle B^i. Let $S_{(i,j)}$ be the jth element of B^i. For each k, segment S_k is assigned to $S_{(i,j)}$, where $k = i \times N + j$:

(a) If $1 \le i < N-1$, $b_i = \lceil S/N \rceil$. In Figure 4, B^i $(1 \le i < 3)$ consists of 4 segments in order.

(b) Otherwise, $b_i \le \lceil S/N \rceil$. In Figure 4, the final segment bundle B^3 consists of balanced segments.

Step 2 (attaching the hash value of the next segment). For each k $(1 \le k < S)$, it computes $H(S_{(k+1)})$ and then concatenates the computed hash value to $S_{(k)}$. Let $S'_{(k)} = S_{(k)} \parallel H(S_{(k+1)})$. If $k = S$, $S'_{(k)} = S_{(k)} \parallel$ null_padding.

Step 3 (constructing MHT). For each i, it computes $H(S'_{(i,1)})$ and then assigns the computed hash value to a leaf node of MHT as its node value in order. Also, it computes the witness W_i of $S'_{(i,1)}$. Finally, it signs the V_1 of MHT with its private key SK. Let the generated signature value be sign.

Step 4 (generating $D_{(k)} = D_{(i,j)}$). For each $j > 1$, let $\text{sign}_{(k)}$ be the signature value of $H(S_{(i,j)})$ generated with SK. The publisher generates $D_{(i,j)}$ which is data for delivering a segment $S_{(i,j)}$ as follows:

(a) If $j = 1$, $D_{(i,1)} = \{S'_{(i,1)}, W_i, \text{sign}\}$.

(b) If $1 < j < b_i$, $D_{(i,j)} = \{S'_{(i,j)}, \text{sign}_{(k)}\}$.

(c) In the case of $j = b_i$, if $k = N$, $D_{(i,j)} = \{S_{(i,j)}, \text{padding}, \text{sign}_{(k)}\}$. Otherwise, $D_{(i,j)} = \{S'_{(i,j)}, \text{sign}_{(k)}\}$.

The $\text{sign}_{(k)}$ attached to $D_{(i,j)}$ is an optional field considering packet loss situation. Hence, if packet loss rate is negligible or the impact of packet loss is not serious, $\text{sign}_{(k)}$ can be removed from $D_{(i,j)}$.

4.2.2. Data Verification. When receiving $D_{(k)} = D_{(i,j)}$, a user verifies the $S_{(i,j)}$ of $D_{(i,j)}$ as follows.

Case 1 ($k = 1$). If $D_{(i,j)}$ is for the first segment, that is, $D_{(i,j)} = D_{(0,1)}$, the user computes V_1 using both S_1 and W_1 and then verifies the sign of $D_{(0,1)}$. If valid, it regards $D_{(0,1)}$ as valid data and then temporarily saves both $H(S_{(0,2)})$ packaged in $D_{(0,1)}$ and the computed V_1 to verify the next data, $D_{(0,2)}$, and $D_{(r,1)}$, respectively.

Case 2 ($k > 1$ and $H(S_{(k)})$ has been saved). If $H(S_{(k)})$ has previously been saved when handling $D_{(k-1)}$, the user computes the hash value of $S_{(k)}$ packaged in $D_{(k)}$ and then compares the computed hash value with the saved $H(S_{(k)})$. If the two values are the same, the user regards $D_{(k)}$ as valid data. Then the user temporarily saves $H(S_{(k+1)})$ packaged in $D_{(k)}$ to verify the next data, $D_{(k+1)}$.

Case 3 ($k > 1$ but $H(S_{(k)})$ has not been saved). (a) If $j = 1$, the user computes V_1 using W_i and then compares the computed V_1 with the previously saved V_1 in Case 1. If the two values are equal, the user regards $D_{(i,1)}$ as valid and then temporarily saves $H(S_{(k+1)})$ packaged in $D_{(k)} = D_{(i,1)}$ to verify the next data, $D_{(k+1)} = D_{(i,2)}$, if $D_{(k)}$ is not the final segment of content.

(b) Otherwise, the user verifies $\text{sign}_{(k)}$ attached in $D_{(k)}$. If valid, it temporarily regards $D_{(k)}$ as a valid data and then saves

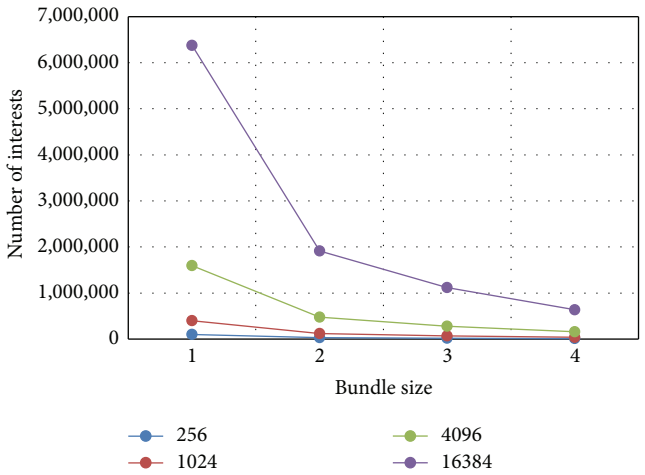


Figure 5: Transmission overheads for forwarding Interest.

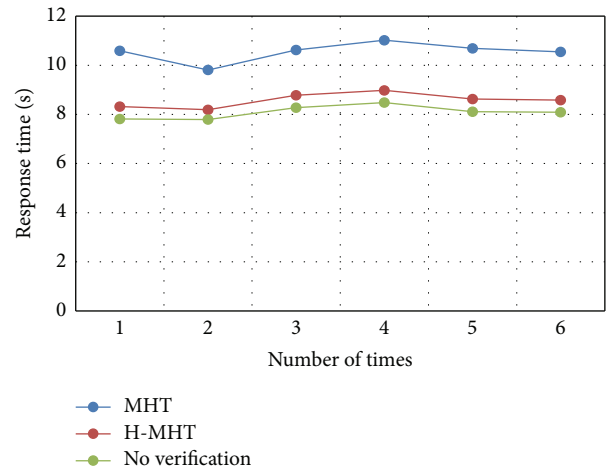


Figure 6: A comparison of response times to share a content fragmented into 256 segments between two smart phones.

both $H(S_{(k)})$ and $H(S_{(k+1)})$. The $D_{(k)}$ will finally be verified after achieving the verification of $D_{(k-1)}$ and comparing the saved $H(S_{(k)})$ with the $H(S_{(k)})$ attached in the valid $D_{(k-1)}$.

5. Performance Evaluation

5.1. Group-Interest Performance. To evaluate the transmission overheads of a Group-Interest, we assume the following:

(i) There are 5 networks connected by 5 border gateways. Each network has a binary tree-shaped network topology with depth 3 consisting of multiple CCN routers. Each end-user is initially placed and then is connected to CCN router, respectively.

(ii) During this simulation, a user utilizes only Group-Interests for requesting content and keeps trying to send *Interests* at a predefined sending rate.

(iii) There are 100 content files which users can access. Each content consists of N (= 256, 1024, 4096, and 16384) segments including meta-data. Each bundle consists of m (= 1, 4, 8, and 16) segments in order.

Then we measure the total amount of transmitted Interest.

Figure 5 shows two results. First, if a bundle size is 1, it means a Group-Interest is actually a general Interest. So when using a Group-Interest ($m > 1$), the transmission overheads of CCN can meaningfully be reduced. Second, the larger the size of the bundle of segments is, the more the transmission overheads of Interest are improved. That is, a Group-Interest is especially efficient when being applied to large size content. However, even if some segments requested by a Group-Interest have been responded, the Group-Interest is continuously forwarded until all requested segments are retrieved. Hence, when utilizing a Group-Interest with size m, transmission performance is not enhanced proportionally to m.

5.2. Content Verification Performance. To analyze the performance of the proposed content verification scheme, we assume the configuration of simulation as described in the previous evaluation. Also, H-MHT and MHT use a binary tree with 8 and 64 leaf nodes, respectively. And we use general Interests, not Group-Interests. Then we measure the time for retrieving content. As shown in Figure 6, the response time is improved by about 20%.

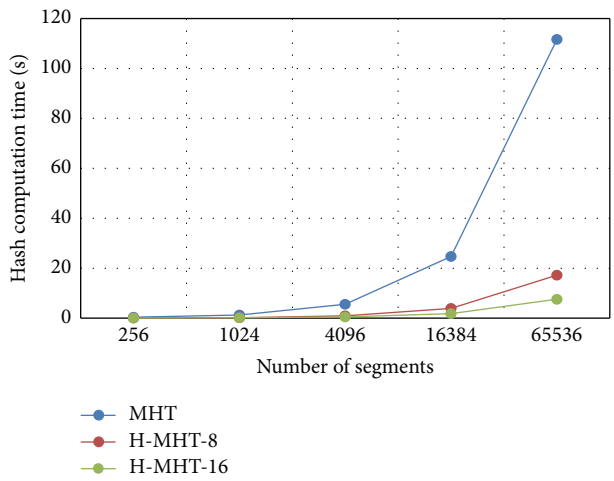


Figure 7: The computation overhead of H-MHT.

Also, we measure the computation overheads of computing hash values for verifying content. Figure 7 shows results considering the cases in which content is fragmented into 256, 1024, 4096, 16384, and 65536 segments, respectively. Then we measure the average time required to compute all hash values for verifying the content:

(i) [MHT] shows the computation overhead of the case of bundle size 1.

(ii) [H-MHT-n] show the results of the cases of bundle size n.

As shown in Figure 7, the larger the bundle size is (as well as the more segments the content is fragmented into), the more efficient the communication overhead is.

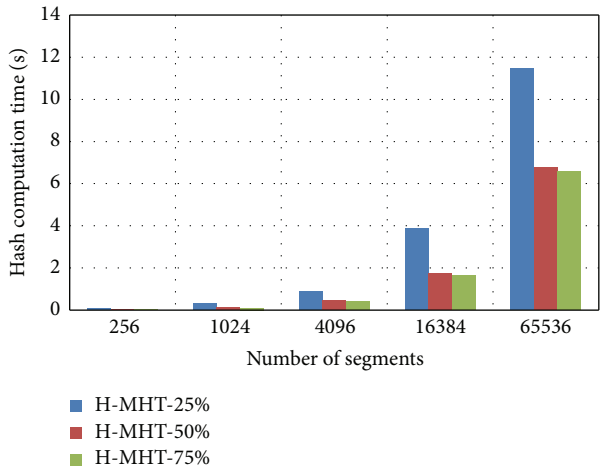


Figure 8: The computation overhead considering different bundle sizes.

Figure 8 is the result of performance measurement of H-MHT considering different bundle sizes. For that, we assume that content is fragmented into 256, 1024, 4096, 16384, and 65536 segments. Let the number of segments be 2^n. For each case, we consider the bundle size as $2^{n\times0.25}$, $2^{n\times0.5}$, and $2^{n\times0.75}$, respectively. As shown in Figure 8, when using the case of a bundle size $2^{n\times0.5}$, the computation efficiency of computing hash values is dramatically improved. But it becomes less effective when using larger bundle sizes than $2^{n\times0.5}$.

6. Conclusion

This paper makes two main points to enhance the performance of CCN. First, since CCN is designed as a request-driven communication model and utilizes fragmented content segments, when requesting content, a user should generate a number of similar Interests to retrieve the content. Using a Group-Interest, it is possible to reduce transmission overheads for forwarding Interests.

Second, an enhanced content verification process is proposed to reduce service latency due to the content verification process of CCN. For that, it is proposed to utilize hash chains. However, when applying a hash chain, it should be considered how to handle packet loss situation. So we also use both MHT and the signature of each segment. Combining MHT to hash chains, it is possible to reduce the computation overheads of a content verification process as well as to limit the effect of packet loss situation. Also, the proposed scheme is designed as being suitable for a Group-Interest. The proposed scheme can provide improved service scalability and low computation costs by reducing the number of hash operations.

These features are important in mobile consumer environments since most mobile consumer devices inherently have limited resource capability. Specially, since various IoT services utilize thin devices like a sensor, these features are meaningful to such services.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported in part by NRF, Republic of Korea, under Grant no. NRF-2013R1A1A2008389.

References

[1] D. Clark, "The design philosophy of the DARPA internet protocols," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 106–114, 1988.

[2] Visual Networking Index (VNI), *Cisco Visual Networking Index: Forecast and Methodology, 2014–2019*, Cisco White Paper, 2015.

[3] L. Y. Huanga, Y. J. Hsiehb, and Y. C. Wu, "Gratifications and social network service usage: the mediating role of online experience," *Information Management*, vol. 51, no. 6, pp. 774–778, 2014.

[4] L. Delgrossi and T. Zhang, *Vehicle Safety Communications*, John Wiley & Sons, 2012.

[5] A. Feldmann, "Internet clean-slate design: what and why," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, 2007.

[6] R. H. Weber, "Internet of things—new security and privacy challenges," *Computer Law & Security Review*, vol. 26, no. 1, pp. 23–30, 2010.

[7] J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 26–36, 2011.

[8] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[9] G. Xylomenos, C. N. Ververidis, V. A. Siris et al., "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[10] K. Pentikousis, B. Ohlman, D. Corujo et al., "Information-centric networking: baseline scenarios," RFC 7476, March 2015.

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '09)*, pp. 1–12, Rome, Italy, December 2009.

[12] V. Jacobson, R. L. Braynard, T. Diebert et al., "Custodian-based information sharing," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 38–43, 2012.

[13] C. Park, T. Kwon, and Y. Choi, "Scalability problem for interest diffusion in content-centric network," in *Proceedings of the 14th Conference on Next Generation Communication Software (NCS '10)*, Pyeongchang, Republic of Korea, December 2010.

[14] S. Choi, K. Kim, S. Kim, and B.-H. Roh, "Threat of DoS by interest flooding attack in content-centric networking," in *Proceedings of the 27th International Conference on Information Networking (ICOIN '13)*, pp. 315–319, Bangkok, Thailand, January 2013.

[15] C. A. Shue, M. Gupta, and M. P. Davy, "Packet forwarding with source verification," *Computer Networks*, vol. 52, no. 8, pp. 1567–1582, 2008.

[16] G. Ma and Z. Chen, "Comparative study on CCN and CDN," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '14)*, pp. 169–170, IEEE, Toronto, Canada, May 2014.

[17] C. Ghali, A. Narayanan, D. Oran, G. Tsudik, and C. A. Wood, "Secure fragmentation for content-centric networks," in *Proceedings of the IEEE 14th International Symposium on Network Computing and Applications (NCA '15)*, pp. 47–56, Cambridge, Mass, USA, September 2015.

[18] M. Amadeo, C. Campolo, and A. Molinaro, "Internet of things via named data networking: the support of push traffic," in *Proceedings of the IEEE International Conference and Workshop on the Network of the Future (NOF '14)*, pp. 1–5, Paris, France, December 2014.

[19] R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '80)*, pp. 122–134, Oakland, Calif, USA, April 1980.

[20] R. J. Bayardo and J. Sorensen, "Merkle tree authentication of HTTP responses," in *Proceedings of the Special Interest Tracks and Posters of the14th International World Wide Web Conference (WWW '05)*, pp. 1182–1183, ACM, May 2005.

[21] K. Ren, W. Lou, K. Zeng, and P. J. Moran, "On broadcast authentication in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4136–4144, 2007.