IEICE TRANS. COMMUN., VOL.E99–B, NO.12 DECEMBER 2016

PAPER Special Section on Information Centric Networking: Paradigms, Technologies, and Applications

List Interest: Simply Packing Interests Dramatically Reduces Router Workload in Content-Centric Networking*

Jun KURIHARA[†a)], **Kenji YOKOTA**[†b)], *and* **Atsushi TAGAMI**[†c)], *Members*

SUMMARY *Content-centric networking* (CCN) is an emerging networking architecture that is being actively investigated in both the research and industrial communities. In the latest version of CCN, a large number of interests have to be issued when large content is retrieved. Since CCN routers have to search several tables for each incoming interest, this could cause a serious problem of router workload. In order to solve this problem, this paper introduces a novel strategy of "grouping" multiple interests with common information and "packing" them to a special interest called the *list interest*. Our list interest is designed to co-operate with the *manifest* of CCN as its dual. This paper demonstrates that by skipping and terminating several search steps using the common information in the list interest, the router can search its tables for the list interest-based request with dramatically smaller complexity than the case of the standard interest-based request. Furthermore, we also consider the deployment of list interests and design a novel TCP-like congestion control method for list interests to employ them just like standard interests.

key words: *content-centric networking, CCNx 1.0, interest aggregation, list interest, manifest*

1. Introduction

Information-centric networking architectures (ICNs) [7], [11] are quickly becoming an attractive alternative to the current host-to-host Internet design in both the research and industrial communities. Several novel networking architectures [1]–[5], [13], [14], [22], [23] have recently been proposed as instances of the ICN. The most common and fundamental features in these ICN instances are: (1) *interest-based content retrieval*, (2) *content oriented naming and routing at the network layer*, and (3) *in-network caching*. Features (1) and (2) imply that users acquire content *from the network* via explicit queries for uniquely named content, rather than by establishing point-to-point connections between endpoints. In-network caching permits a router to cache any content for predetermined lengths of time such that subsequent requests for the same content can be satisfied from the cache, rather than by forwarding the interest upstream. These architectural features enable many foreseeable benefits, such as improved performance [19] and lower network cost [24].

Content-centric networking (CCN) [1], [13], [22] is one of the ICN instances that are being actively investigated. The CCN communication is based on *named data objects* and driven by content clients (called *consumers*); A consumer requests a named data object via an *interest* packet, then any network node that has the requested object satisfies the interest by responding with the named data object called *content object* packet. When a router receives an interest, it first looks up the interest-specifying name in its cache (called the *content store* (CS)) by a certain matching-rule. One interest basically matches only one content object. If a router receiving the interest does not have the requested content object, it then looks up the name again in the *pending interest table* (PIT) that is the stack of unsatisfied interests at the router. If the interest is not found even in the PIT, the router simply forwards the interest upstream according to the routing information in the *forwarding information base* (FIB).

In the early stage of ICN research, most of CCN prototype software and specifications had been implemented and designed as overlay ones as the proof-of-concept. Currently, in order to maximize the CCN benefit in the real world and eliminate the overhead for the overlaid implementation, researchers are actively investigating how to realize the concept of CCN as a native L3 protocol. The newest versions of CCN (CCNx [1] and NDN [3][†]) is designed so as to be used as an L3 protocol over Ethernet, i.e., an alternative to the Internet Protocol (IP). In the CCN as an L3 protocol, the size of content objects should be less than the maximum transmission unit (MTU) of the L2 protocol, say 1500 bytes over Ethernet, in order to avoid the message fragmentation. This could cause a serious problem, especially in cases where a large content, e.g., video, audio, etc., is conveyed over CCN by being split into a large number of small content objects. In such a case, the consumer has to issue the same number of interests as the content objects, and routers may suffer from the heavy computational workload to look up a large number of names of incoming interests in the search space of FIB/PIT/CS entries.

The aim of this paper is to provide a solution to the problem of router workload in the CCN stated above. To this end, we propose a novel strategy of "grouping" multiple interests with common information and "packing" them into a special interest called the *list interest*. The key idea of our list interest is to skip and terminate FIB/PIT/CS look-up operations for multiple interests that are forwarded to

Manuscript received April 8, 2016.
Manuscript revised July 15, 2016.

[†]The authors are with the KDDI R&D Laboratories, Inc., Fujimino-shi, 356-8502 Japan.

*The material in this paper was presented in part at the IEEE MASS 2015 Workshop on Content-Centric Networking (CCN 2015) [16].

a) E-mail: kurihara@ieee.org
b) E-mail: ke-yokota@kddilabs.jp
c) E-mail: tagami@kddilabs.jp

DOI: 10.1587/transcom.2016CNP0004

[†]They branched off from the original CCN project in 2013.

the same destination by packing them using the common information. Our analysis in this paper clarifies that the skip and termination using list interests dramatically reduces the router workload, and hence this could be essential for the future deployment of the CCN. The list interest is also designed to be generated directly from the *manifest* [22] that is a new network message introduced in CCNx [1], and it can be, in other words, viewed as a dual of the manifest since both are structured as catalogs of names. We also consider and discuss the deployment of list interests, and introduce a novel congestion control strategy to control the flow for the list interest based on the TCP-like congestion window. This shows a clear direction to employ list interests just like standard interests.

We should note that the earlier version of this paper [16] proposed only a method for skipping FIB searches. In addition to skipping FIB searches, this paper newly introduces novel methods to terminate CS and PIT searches by list interests. Also, note that in this paper, we shall present the design and description of list interests in the framework of CCNx, especially its latest version called *CCNx 1.0* [1], [22]. Although our presentation in this paper is tailored towards CCNx 1.0, we note that our concept of list interests is compatible with any CCN realizations, e.g., NDN [3].

The rest of this paper is organized as follows. Section 2 briefly introduces CCNx 1.0, and summarizes existing studies related to our work. Section 3 proposes the list interest and describes how to process it at CCN routers. Section 4 shows that the router workload in CCN can be dramatically reduced by introducing the list interest. Section 5 discusses the congestion control mechanism for the list interests. Section 6 finally provides our concluding remarks.

2. Preliminaries

This section first presents a brief introduction of the CCNx 1.0 [13], [22] and then describes its new network message called the *manifest*. Next, we introduce existing studies of the aggregation of multiple interests, which is related to our work.

2.1 Overview of CCNx 1.0

Content-centric networking (CCN) [13], [22] is a future networking architecture, and CCNx 1.0 [1], [22] is the latest protocol design of CCN. There are two basic parties in CCN: (1) *content publishers* and (2) *content consumers*. For the sake of simplicity, we may refer to these parties as publishers and consumers, respectively. A publisher creates named network objects (called *content objects*) with an associated cryptographic signature from any type of content, and publishes them by name over the network. A consumer issues request messages (called *interests*) by publisher-given name, and retrieves content objects that have *exactly the same names* in the requests. This fundamental rule of matching between names and content objects is called the *exact-match*. CCNx 1.0 allows interest messages to carry additional restrictions

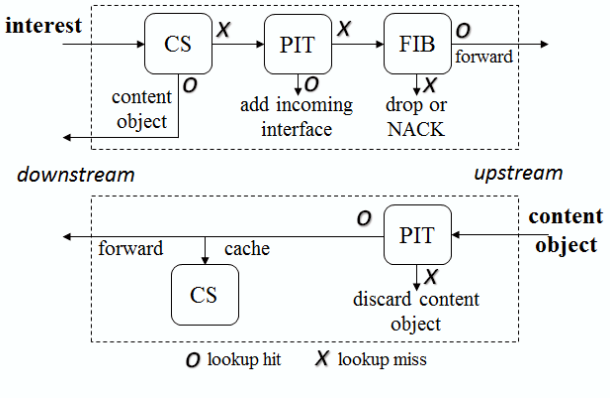


Fig. 1 CCN interest/content object forwarding process (This figure was originally given in [25]).

that are used to determine which content objects may satisfy (match) the interests [17]. In particular, the *hash restriction* of a content object specified in an interest only allows the network to return the content object whose cryptographic hash exactly equals the indicated value.

CCN routers in the network are intermediate nodes that forward interests from consumers and content objects from publishers. They are composed of three primary elements: (1) a forwarding information base (FIB), (2) a pending interest table (PIT) and (3) a content store (CS). The FIB is used to route incoming interests to the appropriate output interface towards the desired content publisher. Much like traditional IP routing tables, the FIB is populated using standard routing protocols or static routes and matches interest names to FIB entries using the *longest-prefix-match*. The PIT serves as a cache of the interest state such that content objects that satisfy interests may follow the reverse interest path back to the consumer. The CS is a cache of content objects that have been processed in case they are re-requested. When an interest comes to a router, it searches its CS for the name in the interest prior to forwarding interests upstream. These caches serve to reduce the latency of retrieving content objects in the network. Figure 1 summarizes the forwarding processes of an interest and a content object via these three components.

2.2 Manifests in CCNx 1.0

The *manifest* is a new type of content object that has been introduced in CCNx 1.0 [22] in order to bundle names of all content objects constituting a large content and provide them in a bulk. A manifest provides meta information about a collection of successive content objects, and enumerates the ordered and hash-based names of every content object of the collection[†].

In the payload of a manifest, the meta information and ordered names are specified with two fields, `ListOfNames` and `ListOfHashes`. The `ListOfNames` field enumerates name entries that indicate the content

[†]See [15] for the detailed format of manifests.

name prefix (MediaName) and its first chunk number (StartChunk). By the ListOfNames field, the manifest can carry multiple collections of content object names and hashes, i.e., it corresponds to multiple large contents. In the ListOfHashes, each entry indicates a content object by a pointer (NameIndex) to its corresponding entry in ListOfNames and the hash (Hash). In other words, the NameIndex points out the line number of ListOfNames in a manifest. To illustrate this binding, consider the case where the first and second entries of the ListOfNames are

```
{StartChunk = 3, MediaName = lci:/obj1},
{StartChunk = 1, MediaName = lci:/obj2},
```

and the first to fourth entries of ListOfHashes are

```
{NameIndex = 1, Hash = 0x1234},
{NameIndex = 1, Hash = 0xABCD},
{NameIndex = 2, Hash = 0x5678},
{NameIndex = 2, Hash = 0xEF01},
```

respectively. Then, the first entry of the ListOfHashes represents a content object with name lci:/obj1/chunk=3 and hash 0x1234, and the second entry represents one with the name lci:/obj1/chunk=4 and hash 0xABCD. Similarly, the third and fourth entries represent lci:/obj2/chunk=1 with hash 0x5678 and lci:/obj2/chunk=2 with hash 0xEF01, respectively. Our list interest is basically generated from the manifest at a consumer and issued as requests for successive content objects listed in the manifest.

2.3 Related Works on Aggregation of Interests

For the older version of the CCN (CCN 0.x), Byun et al. [8] introduced a novel method to aggregate multiple interests with successive names, i.e., chunk number, and create a special interest specifying the range of chunk numbers at consumers. The purpose of their modification was to simply reduce the upload traffic by aggregating multiple interests into one special interest. Although their objective was different from ours (router workload reduction), this approach of aggregating interests was fundamentally similar. However, we may not be able to apply their approach to the latest version of the CCN –CCNx 1.0–. This is because in CCNx 1.0, routers may use only content object hashes to search CS and PIT entries for incoming interests [18], and then names are used just for routing to the content location, namely the FIB search. Hence, in such a case, the consumer is required to specify hashes in addition to routable names in the interests. Thus, since their approach only specifies the range of chunk numbers, it may not work over CCNx 1.0. In Sect. 3, we solve this problem and present the *list interest* designed for CCNx 1.0.

On the other hand, congestion control has to be introduced even for CCNx 1.0 using list interests. Saucez et al. [20] introduced a TCP-like mechanism for the standard CCN based on the congestion window that works in an additive increase multiple decrease (AIMD) manner [12]. Their mechanism controls the number of content objects to arrive by changing the size of the congestion window, i.e., the timing of the issuance of interests, in an adaptive manner. In the case of interest aggregating multiple ones, we have to control the window size according to the number of aggregated interests. Byun et al. proposed a congestion control mechanism [8] for the interest aggregating multiple ones in the CCN 0.x. Their mechanism was designed in such a way that the congestion window size was fixed from the beginning of the communication, and hence it did not control the window size in an adaptive manner relative to the congestion unlike TCP. From this observation, we consider a new window-based congestion control method in Sect. 5.2.1 that can control the window size in an adaptive manner.

3. Reduction of Router Workload by Packing Interests Using Common Information

One considerable drawback of CCNx 1.0 is that consumers may have to issue an enormous number of interests in order to retrieve a content of large size, and then routers may suffer from high computational workloads to process them. The *list interest* is designed to reduce the search complexity at routers for multiple incoming interests with successive names. In the following, we describe the structure of the list interest, its forwarding rule, and its processing rule at routers in the CCNx 1.0. In Sect. 4, we shall estimate how much the computational workload can be reduced by list interests.

3.1 Structure of List Interests and Packing Strategy

Basically, the list interest can be viewed as a catalog of multiple pairs, name and content object hash, of interests. We should first note that a list interest is generated from multiple interests, called *source interests*, by a consumer, where their names have to be the same except for chunk numbers suffixed.

Figure 2 illustrates the message structure of the list interest. In the list interest, we place two new fields in its optional header, MediaName and NumHashEntry. The MediaName field has the prefix that is the longest one included in all the source interests, namely, the name excluding the chunk number. As shown in Fig. 2, the NumHashEntry has the nested structure containing a chunk number of the content object and its hash. For instance, consider the case where pairs of (name, hash of corresponding content object) of source interests are

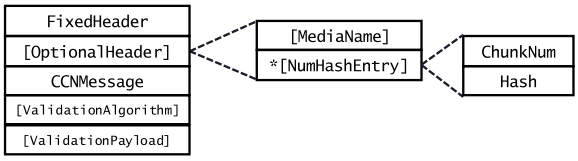


Fig. 2 Structure of list interest, where each field name is given in the CCN specification [17]: $[\cdot]$ means an optional field, $*\cdot$ means zero or more repeated fields.

(lci:/obj1/chunk=3, 0x1234),

and

(lci:/obj1/chunk=4, 0xABCD).

Then, we have

MediaName=lci:/obj1,

and two NumHashEntry are

{ChunkNum=3, Hash=0x1234},

and

{ChunkNum=4, Hash=0xABCD}.

Here we explain a strategy to pack source interests in a list interest and name the list interest. First, source interests that are to be packed in a list interest should be chosen so that their names are mutually different only in the chunk number, as described at the beginning of this subsection. Second, the list interest has to be assigned a name with the same prefix as the MediaName. These are because the list interest has to be routed to the original destination of its source interests. Note that since list interests are defined as kinds of interests, they are routed in exactly the same manner as standard interests by their names.

The packing operation at consumers should be done through manifests explained in Sect. 2.2. As we can see in Fig. 2, the list interest has been designed in such a way that the MediaName and NumHashEntry are directly given from the ListOfNames and ListOfHashes of the manifest with no violation of the packing strategy stated above. We thus see that the list interest can be viewed as a dual of the manifest and that list interests can easily be generated from the information in the manifest.

Here we give an example to generate a list interest directly from a manifest. Suppose that a consumer retrieved a manifest that has the following ListOfNames and ListOfHashes.

```
ListOfNames = {
  {StartChunk=3, MediaName=lci:/obj1}
},
ListOfHashes = {
  {NameIndex=1, Hash=0x1234},
  {NameIndex=1, Hash=0xABCD},
  {NameIndex=1, Hash=0x1A2B},
}.
```

From this manifest, the consumer generates a list interest of the following values, where each of them corresponds to each item of ListOfNames in the manifest.

```
   MediaName = lci:/obj1,
NumHashEntry = {ChunkNum=3, Hash=0x1234},
NumHashEntry = {ChunkNum=4, Hash=0xABCD},
NumHashEntry = {ChunkNum=5, Hash=0x1A2B}.
```

For list interests, we have to determine their names using their MediaName's. For example of the above case, the name of the list interest could be lci:/obj1/chunk=listed.

Consider the case where the consumer wants to leverage multiple paths, i.e., faces, simultaneously to retrieve multiple content objects of the same prefix, i.e., MediaName. Then, list interests for a content should be generated independently for each face. In particular, the consumer first assigns source interests to each face simply by following the basic rule of the standard interest issuance utilizing multiple faces. The consumer then generates a list interest for each face from its assigned source interests. We see that this obviously leverages multiple paths in exactly the same manner as the standard interests.

3.2 How to Process List Interests at Routers

3.2.1 Basic Procedure with Skipping FIB Search

The basic forwarding process for a list interest can be regarded as the *parallelized* one for the normal interest using the MediaName of the list interest. Figure 3 illustrates the process of list interests at routers, and each step in Fig. 3 is described as follows.

(Step 1) A router receives a list interest by discovering its name suffix, e.g., chunk=listed, and/or the existence of MediaName and NumHashEntry in the optional header. It then first reads the optional header of the list interest and extracts full-names and the corresponding content object hashes packed in it.
(Step 2) Then, the router searches CS and PIT by the extracted, say, L pairs of (name, hash) as standard L consecutive look-ups of incoming interests.
(Step 3) Finally, the router updates the list interest by removing the NumHashEntry's that have been look-up hits at CS or PIT, and looks up the FIB entry for the list interest as well as the standard interest.

We can see that in the above flow, although the router looks up multiple (L) entries in CS and PIT, it checks FIB only once. In other words, it skips $L-1$ FIB searches for L consecutive requests. Hence, we can easily expect that this skip dramatically reduces the router workload.

The list interest itself should be discarded by the router when it has no NumHashEntry (all NumHashEntry's are deleted) in Step 3. Note that since routers on the path from the consumer to the router did not create PIT entries for the list interest itself, we can discard it with no violation of the CCN forwarding rule.

Note that since the optional header is not included in the source of validation data [22], they do not affect the validation of the container, i.e., the list interest itself. Thus, the update of the optional header does not affect the validation as well.

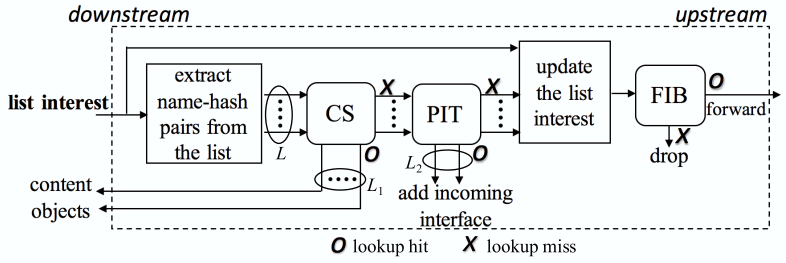


Fig. 3 Brief sketch of router's processing flow of a list interest.

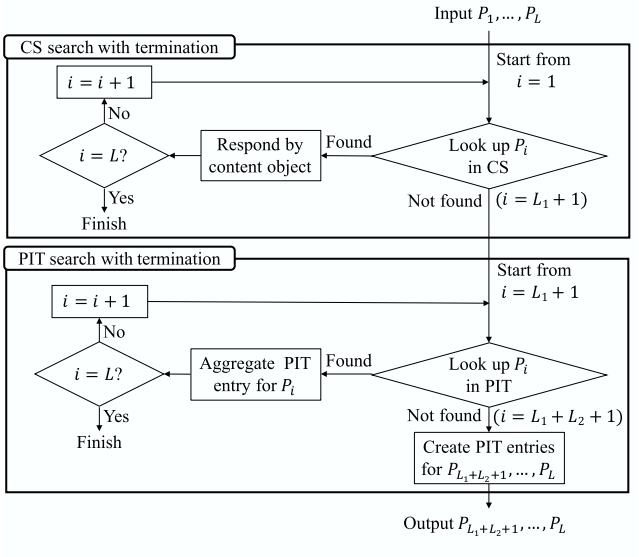


Fig. 4 The flow of CS and PIT searches with termination, which corresponds to (Step 2) in Sect. 3.2.1. The sequence $(P_1, \ldots, P_L)$ represents L (name, hash) pairs extracted from an incoming list interest. After this flow, a new list interest is generated for $(P_{L_1+L_2+1}, \ldots, P_L)$.

3.2.2 Acceleration by Terminating CS and PIT Searches

For L consecutive requests, although the basic forwarding process given in Sect. 3.2.1 can skip $L - 1$ FIB searches, it has to search L times each for CS and PIT. In this subsection, we introduce a pragmatic method to reduce the number of CS/PIT look-ups and accelerate the processing of list interests.

Recall that a list interest is generated from source interests that have successive chunk numbers, and one content is composed of their corresponding content objects. Considering a realistic scenario of content retrieval, we can easily expect that the pending status of a source interest is likely to coincide with that of its previous one at a router. Much the same is true on the caching status of a content object corresponding to a source interest. Based on this observation and expectation, our method terminates queued search operations for source interests at a router when the output of subsequent operations can be expected.

Figure 4 illustrates the flow of our method with termination of CS and PIT searches, which takes L (name, hash) pairs, $P_1, \ldots, P_L$, extracted from a list interest as input. This corresponds to the **(Step 2)** in Sect. 3.2.1. Without loss of generality, we assume that the chunk number of the name in P_i is smaller than that in P_{i+1}. The method consists of two phases, i.e., the CS search phase and PIT search phase, as shown in Fig. 4. In the CS search phase, the router looks up P_i's in CS sequentially from $i = 1$, and immediately terminates the look-up sequence when P_i $(= P_{L_1+1})$ is not found in CS. We suppose it is terminated for $i = L_1 + 1$ as Fig. 3. We then presume that content objects for $P_{L_1+2}, \ldots, P_L$ are not cached, as well as P_{L_1+1}, and start the PIT search phase with $P_{L_1+1}, \ldots, P_L$ as input.

The PIT search phase runs in the same manner as the CS search phase. The router immediately terminates the look-up sequence when P_j $(= P_{L_1+L_2+1})$ is not found in PIT. Then, interests for $P_{L_1+L_2+1}$ and $P_{L_1+L_2+2}, \ldots, P_L$ are regarded as ones that have never been received by the router. The router finally updates the list with $P_{L_1+L_2+1}, \ldots, P_L$ and transitions the phase of the FIB search, i.e., **(Step 3)** of Sect. 3.2.1.

We should note that the termination of CS/PIT search sequences described above can be realized only by listing successive interests, i.e., list interests. If we implement this for standard interests, the router has to always check whether an incoming interest is a part of successive ones for the retrieval of a big content. This should require an undesirable computational workload and extra buffering at routers. On the other hand, list interests themselves explicitly prove that they include successive interests, and hence list interests enables computationally-reasonable operations to terminate successive CS/PIT searches.

As the conclusion of this section, we justify the termination under the 'expectation' on subsequent (name, hash) pairs. Namely, we show that the termination does not violate the forwarding rule even if some erroneous expectations are given. First, consider the case where a content object for a (name, hash) pair P_j is already cached in CS and the CS search is terminated before P_j due to an erroneous expectation of CS entries. A PIT entry for P_j is generated, and it will be satisfied sometime when a content object for P_j newly arrives. Then, the cached content object for P_j will be evicted at some time. Although this could incur a performance loss in terms of throughput, it does not harm the forwarding rule at all. Note that such erroneous expectations of CS entries happen especially after the router evicts CS entries according to its caching strategy. Recall that most caching strategies are mainly based on the popularity of cache entries. We thus

see that only a part of a content, i.e., a part of a sequence consisting of content objects, is rarely evicted. We finally note that since list interests is designed so as to co-exist with standard interests, the caching/pending and eviction rules in CS/PIT should not be customized/specialized for list interests but still follow the standard strategy.

Next consider the case where a PIT entry for P_k is already in PIT and the PIT search is terminated before P_k. Then, a router newly creates another PIT entry for P_k, and issues a new list interest still including P_k. It consequently has two individual PIT entries for P_k. Recall that according to the fundamental matching rule, an issuance of one interest always matches a receipt of one content object at a router. This implies that each of two PIT entries for P_k can be correctly satisfied by each receipt of two individual content objects for P_k. Of course, this incurs unnecessary bandwidth consumption, and may yield the degradation of throughput. In Sect. 5.1, we discuss the probability of erroneous expectations, and present a mathematical analysis that the erroneous expectations are unlikely to happen in the sequential retrieval of content objects composing a large content.

4. Analysis of Look-Up Complexity for List Interests

This section evaluates the complexity to look up FIB/PIT/CS entries via list interests described in the previous section. Consider the case where a router will serve a request for $L(> 0)$ content objects whose names are different only in their chunk numbers. Then, this section first roughly introduces the standard complexity to serve the request given as L individual interests at the CCNx 1.0 router. Next, we estimate the complexity to process the request given by the list interest packing the L interests, and show that the complexity can be dramatically reduced by using the list interest from the standard complexity.

4.1 Router Workload to Process Individual Interests

We first evaluate the complexity of the look-up procedure for one standard interest by referring to the standard process shown in Fig. 1. Figure 5 illustrates the flow of the standard look-up procedure, where each C_* represents the complexity of each step. Table 1 summarizes notations for the complexity in the standard look-up procedure. Here we note that C_{ReadOp} can be regarded as a constant value for any incoming interest, including list interests. Also, note that regardless of the PIT search result, PIT is updated, and hence the PIT search operation always involves the complexity $C_{\mathsf{UpdatePIT}}$.

Considering the router that processes L individual interests of consecutive names, it simply requires L independent executions of Fig. 5. Let $C_{\mathsf{Individual}}$ be the total complexity of the L executions. Let L_1 and L_2 ($L_1 + L_2 \leq L$) be the number of CS look-up hits for the L interests and that of PIT look-up hits, respectively. We then have

Table 1 Notations for complexity of each step in the standard look-up.

Notation	Description
C_{ReadOp}	Parsing and reading the interest
C_{SearchCS}	Searching the CS
$C_{\mathsf{RespondCS}}$	Creating a content object
$C_{\mathsf{SearchPIT}}$	Searching the PIT
$C_{\mathsf{SearchFIB}}$	Looking up the longest matching prefix in FIB
$C_{\mathsf{UpdatePIT}}$	Updating or creating a PIT entry

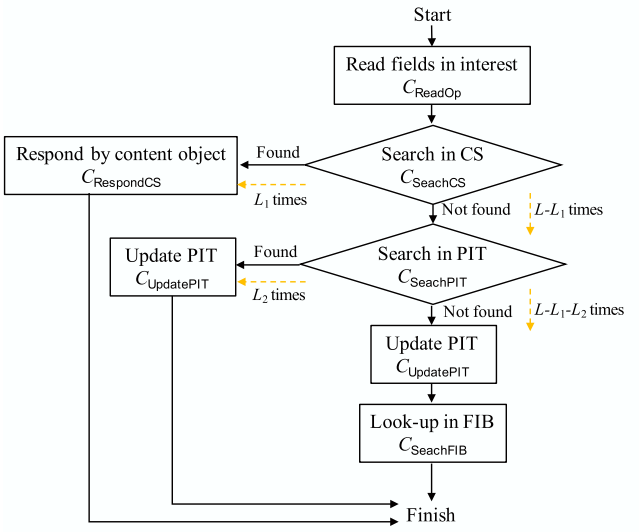


Fig. 5 The flow of the standard look-up procedure for an incoming interest and the complexity of each step, where each C_* represents the complexity of each step (See Table 1).

$$\begin{aligned} C_{\mathsf{Individual}} &= L(C_{\mathsf{ReadOp}} + C_{\mathsf{SearchCS}}) + L_1 C_{\mathsf{RespondCS}} \\ &\quad + (L - L_1)(C_{\mathsf{SearchPIT}} + C_{\mathsf{UpdatePIT}}) \\ &\quad + (L - L_1 - L_2) C_{\mathsf{SearchFIB}}, \end{aligned} \tag{1}$$

where we have assumed that the CS search operation always involves the constant complexity C_{SearchCS}, and that the same is true of $C_{\mathsf{SearchPIT}}$ and $C_{\mathsf{SearchFIB}}$. We can see that L_1 and L_2 highly depend on the popularity of the requested content, and L_1 and L_2 could be positive only for very popular content. If neither CS nor PIT entry is found for each one, the complexity $C_{\mathsf{Individual}}$ is proportional to L.

4.2 Look-Up Complexity for the List Interest

As in the previous subsection, here we summarize the look-up procedure for a list interest, which is shown in Fig. 3. Figure 6 briefly illustrates the flow of the look-up procedure for the list interests packing L successive source interests, and also shows the complexity in each step. In Fig. 6, we have assumed that the read and write (update) operations over an object on a memory have the same number of CPU clock cycles[†]. We thus estimate the complexity to update the list interest as C_{ReadOp}.

First, suppose that the router receives a list interest in

[†]In fact, each of the read and write operations on Pentium processors involves three clock cycles [10].

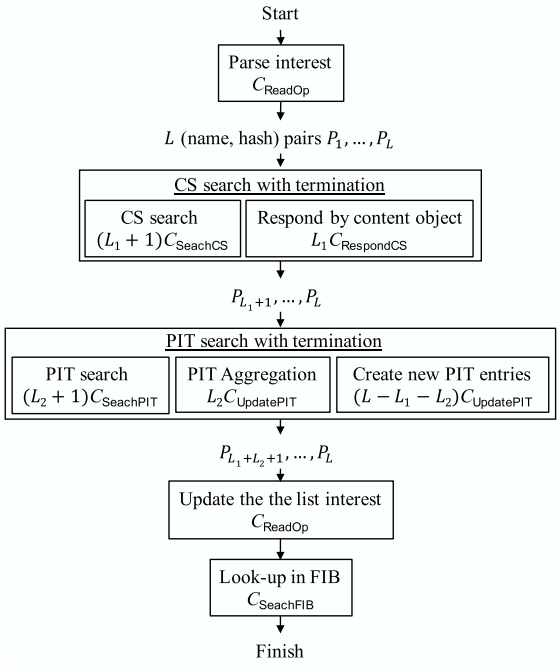


Fig. 6 The brief flow of the look-up procedure for an incoming list interest containing L `NumHashEntry`'s and the complexity of each step, where the CS and PIT the search operation are terminated at $i = L_1 + 1$ and $i = L_1 + L_2 + 1$ as Fig. 4, respectively. (See Fig. 4 for the detailed flow in CS/PIT search with termination.)

which L `NumHashEntry`'s are contained. Also, assume that in the CS and PIT search phases, the search operations are terminated at $i = L_1 + 1$ and $i = L_1 + L_2 + 1$ as Fig. 4, respectively. We should note that when L_1 and L_2 in Fig. 6 exactly equal L_1 and L_2 in Fig. 5 for a fixed set of L source interests, it is the ideal case where we have no erroneous expectation at termination. As shown in Fig. 6, the look-up complexity for L `NumHashEntry`'s in CS and PIT can be given as

$$\begin{aligned}\text{CS: } & (L_1+1)C_{\mathsf{SearchCS}} + L_1 C_{\mathsf{RespondCS}},\\ \text{PIT: } & (L_2+1)C_{\mathsf{SearchPIT}} + (L-L_1)C_{\mathsf{UpdatePIT}},\end{aligned}$$

respectively. From this observation, the total complexity C_{List} in the case of the list interest packing L `NumHashEntry`'s is given as follows.

$$\begin{aligned}C_{\mathsf{List}} =& 2C_{\mathsf{ReadOp}} + (L_1+1)C_{\mathsf{SearchCS}} + L_1 C_{\mathsf{RespondCS}}\\ &+ (L_2+1)C_{\mathsf{SearchPIT}} + (L-L_1)C_{\mathsf{UpdatePIT}}\\ &+ C_{\mathsf{SearchFIB}},\end{aligned} \tag{2}$$

where we have assumed that $L_1 + L_2 < L$.

4.3 Comparison with the Standard Look-Up

Here we compare the total look-up complexity of L individual interests $C_{\mathsf{Individual}}$ with that of one list interest C_{List} containing the L `NumHashEntry`'s of the same interests, where $C_{\mathsf{Individual}}$ and C_{List} are given in Eqs. (1) and (2), respectively.

In the following, we analyze the ratio $C_{\mathsf{List}}/C_{\mathsf{Individual}}$ that represents the reduction rate of the router's workload obtained by the list interest. From now on, we will call L the *list size*.

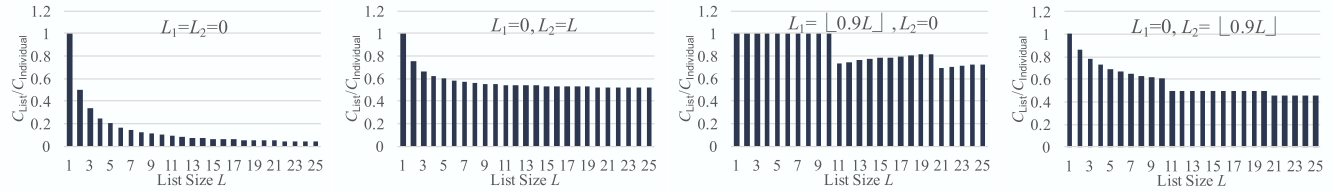
First, we introduce some reasonable assumptions. Since the search on the FIB/CS/PIT must involve a number of memory accesses, we can assume that C_{ReadOp} can be omitted in Eqs. (1) and (2) as a negligible cost. Note that since $C_{\mathsf{UpdatePIT}}$ is the complexity to update/create a PIT entry after the execution of a PIT look-up, it is just a write access to the memory. We thus see $C_{\mathsf{UpdatePIT}} << C_{\mathsf{SearchPIT}}$, and $C_{\mathsf{UpdatePIT}}$ must be negligible compared to $C_{\mathsf{SearchPIT}}$. Similarly, we can also see that the complexity of memory access $C_{\mathsf{RespondCS}}$ is negligible compared to C_{SearchCS}. From these assumptions, we can have the following approximation of $C_{\mathsf{List}}/C_{\mathsf{Individual}}$ that clearly shows the dominant factors in the reduction of the router workload and when the ratio is minimized.

$$\begin{aligned}&C_{\mathsf{List}}/C_{\mathsf{Individual}}\\ &\simeq \frac{(\hat{L}_1+1)C_{\mathsf{SearchCS}} + (\hat{L}_2+1)C_{\mathsf{SearchPIT}} + C_{\mathsf{SearchFIB}}}{LC_{\mathsf{SearchCS}} + (L-L_1)C_{\mathsf{SearchPIT}} + (L-L_1-L_2)C_{\mathsf{SearchFIB}}},\end{aligned} \tag{3}$$

where, in order to clarify the difference, we have supposed that L_1 and L_2 are those in Fig. 5, and that $\hat{L}_1$ and $\hat{L}_2$ are L_1 and L_2 in Fig. 6. We can immediately see from Eq. (3) that if the router have never previously processed corresponding interests and content objects (i.e., the worst case scenario in terms of cache-hit rate), the ratio $C_{\mathsf{List}}/C_{\mathsf{Individual}}$ fundamentally depends only on L, and the power of the list interest is maximized in terms of the reduction of router workload by the skip and termination of FIB/CS/PIT look-ups.

Figure 7 illustrates the ratio $C_{\mathsf{List}}/C_{\mathsf{Individual}}$ for various settings of L, L_1 and L_2. In Fig. 7, we have supposed the following settings for the list size L. First suppose that the size of the interest is limited to 1280 bytes. Then, the maximum possible size of the optional header of interest packets is less than 1260 bytes (8 bytes fixed header and at least 12 bytes CCN message). Recall that the hash restriction of the content object is computed by SHA-256 [17]. Thus, we also assume that the `MediaName` and each `NumHashEntry` are 16 bytes and 48 bytes in the TLV format [17]. From these assumptions, the maximum possible list size is limited to $L = 25$ in Fig. 7. In Fig. 7, we have also assumed that $L_1 = \hat{L}_1$, $L_2 = \hat{L}_2$, $C_{\mathsf{SearchCS}} = C_{\mathsf{SearchPIT}}$ and $3.08C_{\mathsf{SearchCS}} = C_{\mathsf{SearchFIB}}$[†]. Namely, we assumed an ideal environment for list interests where no erroneous expectations occur. We should note that such an ideal environment is also realistic for the retrieval of successive content objects of a large content. As we can see in Fig. 7, the reduction rate of the complexity is $C_{\mathsf{List}}/C_{\mathsf{Individual}} \leq 1$ in any case, i.e., the

[†]The paper [21] evaluated the average numbers of FIB look-ups (simple search of the bash-based FIB) in one process of their algorithm for different prefix lengths from which the look-up of hash-based FIB starts. 3.08 is the minimum one in their evaluation. Since the average complexity of searching hash tables is $O(1)$, we set $3.08C_{\mathsf{SearchCS}} = C_{\mathsf{SearchFIB}}$ in the example as a rough assumption.



(a) $L_1 = L_2 = 0$ (b) $L_1 = 0, L_2 = L$ (c) $L_1 = \lfloor 0.9L \rfloor, L_2 = 0$ (d) $L_1 = 0, L_2 = \lfloor 0.9L \rfloor$

Fig. 7 The ratio of the complexities of the list interest and that of the standard individual interests $C_{\mathsf{List}}/C_{\mathsf{Individual}}$ for L with various settings of L_1 and L_2, where we have assumed $L_1 = \hat{L}_1$, $L_2 = \hat{L}_2$, $C_{\mathsf{SearchCS}} = C_{\mathsf{SearchPIT}}$ and $3.08C_{\mathsf{SearchCS}} = C_{\mathsf{SearchFIB}}$.

gain of the list interest is always positive, and it can be seen to be approximately inversely proportional to the list size L. Especially in Fig. 7(a), i.e., the worst case scenario for router workload, the list interest aggregating multiple successive interests can reduce the workload of the look-up operations at the router dramatically to at most 4% of the standard individual interests. Observing Figs. 7(a) to (d), we can also see that as L_1 and L_2 increases, the gain of the list interest becomes smaller although it is always positive.

Since the complexities C_{SearchCS}, $C_{\mathsf{SearchPIT}}$ and $C_{\mathsf{SearchFIB}}$ themselves are highly dependent on the network status, memory architectures, databases, etc., we cannot strictly and generally estimate the gain of the list interest here. However, we claim that in general for the small $(L_1 + L_2)/L << 1$, the reduction rate of the complexity by the list interest dramatically improves (becomes smaller) as the list size L increases. This is because of the following. Recall that the number of FIB look-ups for one list interest is always 1 regardless of L_1 and L_2, and that $C_{\mathsf{SearchFIB}}$ is generally more serious for routers than C_{SearchCS} and $C_{\mathsf{SearchPIT}}$ due to the search for the longest matching prefix in FIB. From these facts, we see that *even if we do not execute the termination at PIT and CS searches*, we always have the positive gain by the list interest as L increases. Moreover, observe that we always have $C_{\mathsf{List}}/C_{\mathsf{Individual}} \simeq 1$ for any L when $L_1 = L$ from Eq. (3). This implies that the complexity of processing the list interest can be estimated to be almost the same as that for the standard L interests. Therefore, we conclude that the list interest can reduce the router's workload to process incoming interests in any cases.

5. Discussion

In this section, we shall discuss topics on the implementation and deployment of list interests. First, we consider the probability of erroneous expectation at the termination of CS and PIT searches. We then consider the congestion control for list interests and the list size L for deployment.

5.1 Erroneous Expectation for Termination

Here we mathematically express the probability of erroneous expectations in the termination of CS/PIT searches. We first give several notations. Let $f(P_i)$ be a Boolean random variable meaning the statement "the (name, hash) pair P_i is found in CS." We also let $g(P_i)$ be one meaning "P_i is found in PIT." For the sake of simplicity, we represent the Boolean values true and false by T and F, respectively. For example, $f(P_i) = \mathsf{T}$ represents an *event* where a CS entry is successfully found for P_i, and $f(P_i) = \mathsf{F}$ expresses its inversion.

For the CS search operations with termination, the following equation gives the probability that there exist one or more CS entries for a (name, hash) pair $P_{\hat{L}_1+i}$ $(i > 1)$ after a cache-miss for $P_{\hat{L}_1+1}$, i.e., the expected value $\hat{L}_1$ is smaller than the exact value L_1.

$$\begin{aligned}
&\Pr\left[\hat{L}_1 < L_1\right] \\
&= 1 - \Pr\left[\bigcap_{i=\hat{L}_1+2}^{L} (f(P_i) = \mathsf{F}) \,\middle|\, f(P_{\hat{L}_1+1}) = \mathsf{F}\right] \\
&= 1 - \prod_{k=\hat{L}_1+2}^{L} \Pr\left[f(P_k) = \mathsf{F} \,\middle|\, \bigcap_{j=\hat{L}_1+1}^{k-1} (f(P_j) = \mathsf{F})\right], \qquad (4)
\end{aligned}$$

where we have supposed that for individual events, say $X = x$ and $Y = y$, $(X = x) \cap (Y = y)$ means that both $X = x$ and $Y = y$ simultaneously occur. Similarly, for the erroneous expectation in PIT search, we have

$$\begin{aligned}
&\Pr\left[\hat{L}_2 < L_2\right] \\
&= 1 - \Pr\left[\bigcap_{i=\hat{L}_1+\hat{L}_2+2}^{L} (g(P_i) = \mathsf{F}) \,\middle|\, g(P_{\hat{L}_1+\hat{L}_2+1}) = \mathsf{F}\right] \\
&= 1 - \prod_{k=\hat{L}_1+\hat{L}_2+2}^{L} \Pr\left[g(P_k) = \mathsf{F} \,\middle|\, \bigcap_{m=\hat{L}_1+\hat{L}_2+1}^{k-1} (g(P_m) = \mathsf{F})\right]. \qquad (5)
\end{aligned}$$

Assume that a router has the CS and PIT of sufficiently-large size, and that the eviction of CS entries and the timeout of PIT entries have never been executed. We also assume that the consumer wants to retrieve a content that consists of L consecutive content objects represented by (name, hash) pairs $P_1, \ldots, P_L$, and that P_i exactly corresponds to the i-th content object in the sequence. Assuming that such content objects are queried sequentially in the order $P_1, \ldots, P_L$, we see that the caching status for P_i, i.e., the random variable $f(P_i)$, depends on the previous ones $f(P_{i-1}), f(P_{i-2}), \ldots$. For the sake of simplicity, here we introduce a rough assumption that the random variable $f(P_i)$ depends only on its

previous one $f(P_{i-1})$ and is mutually independent of other random variables. We thus have the following simplified version of Eq. (4).

$$\Pr\left[\hat{L}_1 < L_1\right] = 1 - \prod_{k=\hat{L}_1+2}^{L} \Pr\left[f(P_k) = \mathsf{F} \mid f(P_{k-1}) = \mathsf{F}\right], \quad (6)$$

By introducing the same assumption on the random variable $g(P_i)$, we obtain the simplified Eq. (5) as

$$\Pr\left[\hat{L}_2 < L_2\right] = 1 - \prod_{k=\hat{L}_1+\hat{L}_2+2}^{L} \Pr\left[g(P_k) = \mathsf{F} \mid g(P_{k-1}) = \mathsf{F}\right]. \quad (7)$$

To make Eqs. (6) and (7) simpler for the rough estimation of the probability of erroneous expectations, here we give a supposition that the conditional probability of $f(P_i) = \mathsf{F}$ given $f(P_{i-1}) = \mathsf{F}$ and that for $g(P_i) = \mathsf{F}$ and $g(P_{i-1}) = \mathsf{F}$ as follows.

$$\Pr[f(P_i) = \mathsf{F} \mid f(P_{i-1}) = \mathsf{F}] = \alpha,$$

and

$$\Pr[g(P_i) = \mathsf{F} \mid g(P_{i-1}) = \mathsf{F}] = \beta,$$

for any i, where we have assumed that α and β are constant probabilities. Under these assumptions, we have

$$\Pr\left[\hat{L}_1 < L_1\right] = 1 - \alpha^{L-\hat{L}_1-1},$$

and

$$\Pr\left[\hat{L}_2 < L_2\right] = 1 - \beta^{L-\hat{L}_1-\hat{L}_2-1},$$

from Eqs. (6) and (7), respectively. This implies that for $L \leq 25$ as Fig. 7, we can always obtain sufficiently small probabilities, say $\Pr\left[\hat{L}_1 < L_1\right], \Pr\left[\hat{L}_2 < L_2\right] < 0.05$, when α, β are sufficiently high, e.g., $\alpha, \beta > 0.998$. Here we note that α and β can be viewed as dependencies between $f(P_i)$ ($g(P_i)$) and the previous one $f(P_{i-1})$ ($g(P_{i-1})$). This implicitly means that as the dependency among $f(P_1), \ldots, f(P_L)$ ($g(P_1), \ldots, g(P_L)$) increases, the probability of erroneous expectations decreases. Hence, the above analysis gives the evidence of our key idea that in the sequential retrieval of 'strongly-mutually-dependent' content objects, like those composing videos, erroneous expectations are unlikely to incur in the termination of CS and PIT searches. In other words, the termination should be done only for such large contents.

5.2 Congestion Control for List Interests

Considering the deployment of list interests, we have to use some kind of congestion control method specialized for list interests. However, congestion control methods of CCN are currently under investigation in the research community, and no specific approach is given in any CCN specification and implementation even for standard interest-based content retrieval so far. Hence, in this subsection, we pick up an existing 'TCP-like' approach as a popular instance, and give its novel modification to list interests as a consideration of the deployment. Our method for list interests performs exactly the same as the original method for standard interests in terms of the throughput, and implies a way of the deployment of list interests in the CCN.

Algorithm 1 Our congestion control method utilized for list interests: RTO stands for the retransmission timeout. A variables W is the window size, and another variable P is the number of content objects that a consumer waits for their arrival as responses to interests previously transmitted. The list size L is the predefined constant value in this method. N is the value of the counter used to increment the window size.

```
initialization: W ← L, P ← L, N ← 0
if Receive content object until RTO then
    if Is slow start phase then
        W ← W + 1
    else (congestion avoidance phase)
        N ← N + 1
        if N = W then
            W ← W + 1
            N ← 0
        end if
    end if
else
    W ← max{W/2, L}
    N ← 0
end if
P ← P + 1
while W ≥ P + L do
    Pack L interests into a list interest and send it
    P ← P + L
end while
```

5.2.1 A Window-Based Congestion Control Method

As mentioned in Sect. 2.3, Byun et al. [8] proposed a window-based congestion control method for the interest aggregating multiple ones in CCN 0.x, and their scheme can be applied to the list interest in CCNx 1.0. However, the window size is always fixed in their scheme, which is defined as the number of interests that can be issued without waiting for the responses (content objects) to the packets previously transmitted. This implies that their scheme cannot adaptively control the number of requests for content objects according to the network status unlike TCP.

For the above problem, here we give the TCP-like congestion control method for list interests that can control the window size in the same fashion as TCP. Algorithm 1 describes our congestion control method designed for list interests. The purpose of our method is to control the number of content objects in-flight in the network by controlling the window size adaptively in the exactly the fashion as TCP.

Our method has two phases, i.e., the slow start phase and the congestion avoidance phase, as congestion control methods for TCP. Although the window size W is determined in the same manner as TCP Reno [12] at lines 2–15 of Algorithm 1, we can take other window control methods and determine W in a different fashion. We also see that by introducing lines 2–15, Algorithm 1 can adaptively control the window size W unlike [8]. At lines 17–20, the algorithm packs L interests into a list interest, and the number of content objects to arrive P is updated. We should note that if $L = 1$ in Algorithm 1, it coincides with the existing congestion control method for the standard interest [20]. This implies that Algorithm 1 is a natural extension of the standard method for CCN [20] to the list interest.

5.2.2 Relationship between List Size and Performance

As shown in Sect. 4, the router complexity decreases as the list size L grows. However, we have no strategy to determine L in Algorithm 1 for now, and hence we need to evaluate how the performance of the content retrieval depends on L. To this end, here we simulate the content retrieval using CCN and evaluate the performance for the different L's using Algorithm 1.

Table 2 summarizes the simulation environment, and Figure 8 describes the network topology used in the simulation. In order to evaluate the performance of our algorithm, we use the following metrics: *utilization* and *fairness*.

- *Utilization U*: Utilization U is defined as the amount of transferred data over the bottleneck link during some time interval divided by the product of link capacity and that time interval, as given by

$$U = \frac{(\textit{transferred segments}) \cdot (\textit{segment size})}{(\textit{link capacity}) \cdot \textit{time}}.$$

  We should note that the utilization is generally an important metric, since users can download content in a short time if the utilization is high.
- *Fairness F*: The fairness in this paper follows the simple rule that if there are n flows through a bottleneck link, each flow should account for $1/n$ of the bandwidth of the bottleneck link. In this context, the fairness F, a.k.a., Jain's metric of fairness [9], is defined as

$$F = \frac{(\sum_{i=1}^{n} b_i)^2}{n \cdot \sum_{i=1}^{n} b_i^2},$$

  where n is the number of flows, b_i is the rate of flow i. This metric ranges continuously in value from $1/n$ to 1, with 1 corresponding to equal allocations for all users. We extend this metric to evaluate the fairness for the specific length of periods. The fairness F_τ with a measuring period τ is defined as

$$F_\tau = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{(\sum_{i=1}^{n} b_{ij})^2}{n \cdot \sum_{i=1}^{n} b_{ij}^2},$$

  where m is the total number of measuring period, b_{ij} is the rate of flow i in the jth period.

Table 2 Simulation environment.

Simulator	: ndnSIM [6]
Header size	: 41–44 bytes (varying in ndnSIM)
Payload size of content objects	: 1024 bytes
Queue size of CCN routers	: 100 packets
Initial slow-start threshold	: 64
Simulation time	: 100 s

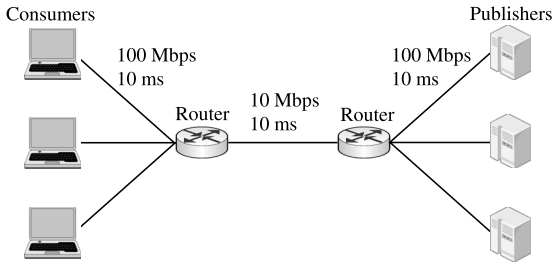
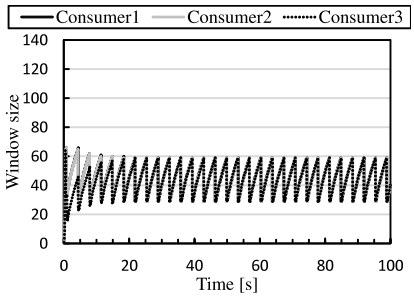


Fig. 8 A dumbbell topology and the setting of propagation delay and capacity for each link.

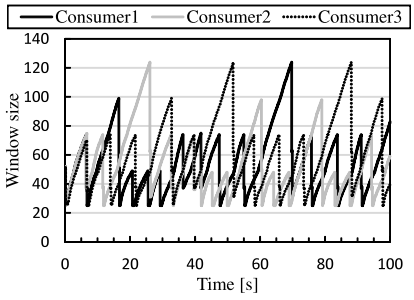
By substituting the result of the simulation into the above definitions of U and F_τ, here we shall evaluate the performance. Table 3 summarizes the simulation result, and Fig. 9 illustrates the time variation of the window size. As we can see in Table 3, the utilization decreases as the list size increases. This is because content objects are intermittently transmitted by using the list interest. However, the difference of the utilization between the normal interest ($L = 1$) and the list interest ($L > 1$) is less than 0.01. Thus, the disadvantage of the list interest is quite small. We can also see in Table 3 and Fig. 9 that the fairness of $\tau = 1$s and 10s for the list interest is lower than 0.9 for any $L > 1$. This is because the list interest generates the burst transmission of content object packets, and accordingly, content object packets in a particular flow are dropped when the router queue is full. On the other hand, the fairness of $\tau = 100$s for the list interest ($L > 1$) is almost the same as the normal interest ($L = 1$) for any L. Considering the case where a large content is retrieved through CCN, the fairness for the long period, i.e., $\tau = 100$s can be more important than that for short period, i.e., $\tau = 1$s and 10s. Hence, we claim that from the results of the utilization and the fairness for $\tau = 100$s, the congestion control for the list interest ($L > 1$) demonstrates almost the same performance as the exiting scheme [20] for the normal interest ($L = 1$). From these results and observations, we thus conclude that in this network setting, the list interest

Table 3 Utilization U and fairness F with each τ for the list size L, where we note that the case of $L = 1$ shows the result of the existing congestion control method [20] for the standard interest.

List size L	1	5	10	15	20	25
U	0.985	0.985	0.983	0.983	0.979	0.978
F_τ (τ=1s)	0.980	0.562	0.493	0.468	0.586	0.646
F_τ (τ=10s)	0.997	0.760	0.688	0.651	0.767	0.847
F_τ (τ=100s)	1.000	0.985	0.971	0.961	0.986	0.995



(a) $L = 1$



(b) $L = 25$

Fig. 9 Time variation of the window size, where we set $L = 1$ and $L = 25$.

with congestion control does not harm the performance of the content retrieval in the CCN for any L. Therefore, we should set L as large as possible, i.e., $L = 25$ in the setting of interests of size 1280 bytes, from the viewpoint of the router workload. Of course, we have to consider the various and realistic cases of a lossy network, a carrier network, etc. in order to find the optimal strategy to determine L in any situation. We remain this as a future work.

Although we considered an end-to-end congestion control method in this section, hop-by-hop ones are also imaginable. For instance, each router counts the number of entries in each flow on PIT and compares the list size to a threshold when it receives the list interest. If the list size exceeds the threshold, the router changes the list interest so as not to exceed the threshold and returns a NACK packet to the consumer. This method possibly improves the fairness of the short period, e.g., $\tau = 1$s, for the list interest. We should note, however, that such a hop-by-hop method involves additional routers' workload in order to control flows at every router, which violates the main objective of list interests.

6. Concluding Remarks

In order to reduce the router workload in CCNx 1.0, this paper proposed the list interest that is generated from multiple interests with their mutually-common information. We demonstrated that by requesting content objects through list interests, routers can search FIB/PIT/CS with dramatically smaller complexity than the case of the standard interest-based request. Furthermore, we also proposed a novel congestion control mechanism for list interests based on TCP-like congestion window. As the conclusion of this paper, we claim that our list interest is a natural consequence of the concept of the manifest in CCNx 1.0, and that we should use list interests as the requests for content objects enumerated in the manifests from the relationship between the list interest and the manifest.

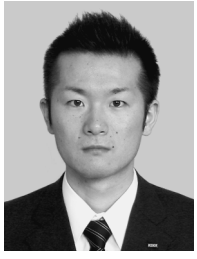
Acknowledgments

The authors would like to thank the editor and two anonymous reviewers for their helpful comments. This work has been supported by the ICN2020 Project (Advancing ICN towards real-world deployment through research, innovative applications, and global scale experimentation), a research project supported jointly by the European Commission under its HORIZON 2020 (Grant Agreement No. 723014) and the National Institute of Information and Communications Technology (NICT) in Japan (Contract No. 184).

References

[1] "The CCNx webpage," http://ccnx.org/
[2] "The FP7 4WARD project," http://www.4ward-project.eu/
[3] "Named-data networking," http://named-data.net/
[4] "Pursuing a pub/sub internet (PURSUIT)," http://www.fp7-pursuit.eu/PursuitWeb/
[5] "Scalable and adaptive internet solutions (SAIL)," http://www.sail-project.eu/
[6] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," Nemed Data Networking, Technical Report, NDN-0005, 2012.
[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," IEEE Commun. Mag., vol.50, no.7, pp.26–36, July 2012.
[8] D. Byun, B.-J.B.J. Lee, and M.-W. Jang, "Adaptive flow control via Interest aggregation in CCN," Proc. 2013 IEEE International Conference on Communications (ICC), pp.3738–3742, 2013.
[9] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," Comput. Networks. ISDN Syst., vol.17, no.1, pp.1–14, 1989.
[10] S.P. Dandamudi, Introduction to Assembly Language Programming: For Pentium and RISC Processors, 2nd ed. Springer, 2006.
[11] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: Seeing the forest for the trees," Proc. 10th ACM Workshop on Hot Topics in Networks, HotNets'11, pp.1–6, 2011.
[12] V. Jacobson, "Congestion avoidance and control," Proc. Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM'88, pp.314–329, 1988.
[13] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, "Networking named content," Proc. 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT'09, pp.1–12, 2009.
[14] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," SIGCOMM Comput. Commun. Rev., vol.37, no.4, pp.181–192, Oct. 2007.
[15] J. Kurihara, E. Uzun, and C.A. Wood, "An encryption-based access control framework for content-centric networking," Proc. 2015 IFIP Networking Conference (IFIP Networking), pp.1–9, 2015.
[16] J. Kurihara, K. Yokota, K. Ueda, and A. Tagami, "List interest: Packing interests for reduction of router workload in CCN 1.0," Proc. 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, pp.500–505, 2015.

[17] M. Mosko and I. Solis, "CCNx messages in TLV format," IRTF ICNRG Internet-Draft, March 2015, available at https://tools.ietf.org/html/draft-mosko-icnrg-ccnxmessages-01
[18] M. Mosko, "Nameless objects," IRTF ICNRG, Jan. 2016.
[19] I. Psaras, R.G. Clegg, R. Landa, W.K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," Proc. NETWORKING 2011, Lecture Notes in Computer Science, vol.6640, pp.78–91, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
[20] D. Saucez, L.A. Grieco, and C. Barakat, "AIMD and CCN: Past and novel acronyms working together in the future Internet," Proc. 2012 ACM Workshop on Capacity Sharing, CSWS'12, pp.21–26, 2012.
[21] W. So, A. Narayanan, and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables," Proc. Architectures for Networking and Communications Systems, pp.215–225, 2013.
[22] I. Solis and G. Scott, "CCN 1.0 tutorial," in ACM ICN 2014, Sept. 2014.
[23] D. Trossen and G. Parisis, "Designing and realizing an information-centric internet," IEEE Commun. Mag., vol.50, no.7, pp.60–67, July 2012.
[24] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," Proc. 21st International Conference on Computer Communications and Networks (ICCCN), pp.1–7, 2012.
[25] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," ACM SIGCOMM Comput. Commun. Rev., vol.42, no.3, pp.62–67, June 2012.



Jun Kurihara received the B.E. degree in computer science, the M.E. degree in communication engineering and the Ph.D. degree in electrical and electronic engineering, all from Tokyo Institute of Technology, Tokyo, Japan, in 2004, 2006 and 2012 respectively. He joined KDDI Corp., Tokyo, Japan in April 2006. Since July 2006, he has been with KDDI R&D Laboratories, Inc., Saitama, Japan as a researcher. From 2013 to 2014, he was also a visiting researcher at Palo Alto Research Center (PARC), Palo Alto, CA, USA. His research interests include coding theory, networking architecture and information security. He received the Best Paper Award from IEICE in 2014.

Kenji Yokota received the B.E. degree in Electrical and Electronic Engineering, the M.E. and Ph.D. degrees in Informatics from Kyoto University, Japan in 2008, 2010, and 2012, respectively. He joined KDDI Corp., Japan in 2012. He is currently a research engineer at Future Communication Architecture Laboratory, KDDI R&D Laboratories, Inc., Japan. His research interests include Information-Centric Networking and ubiquitous computing.

Atsushi Tagami received the M.E. and Ph.D. degrees in Computer Science from Kyushu University, Japan in 1997 and 2010, respectively. He joined KDDI R&D Laboratories Inc. in 1997, where he has been engaged in research and development on Performance Measurement of Communication Networks and Overlay Networking. He is currently a senior manager at Future Communication Architecture Laboratory of KDDI R&D Laboratories, Inc. He received the Excellent Paper Award from IEICE in 2015.