electronics

MDPI

Article

Reliable Multicast Based on Congestion-Aware Cache in ICN

Yingjie Duan [1,2], Hong Ni [1,2] and Xiaoyong Zhu [1,2,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; duanyingjie17@mails.ucas.ac.cn (Y.D.); nih@dsp.ac.cn; (H.N.)

[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China

* Correspondence: zhuxy@dsp.ac.cn; Tel.: +86-131-2116-8320

Abstract: Reliable multicast distribution is essential for some applications such as Internet of Things (IoT) alarm information and important file distribution. Traditional IP reliable multicast usually relies on multicast source retransmission for recovery losses, causing huge recovery delay and redundancy. Moreover, feedback implosion tends to occur towards multicast source as the number of receivers grows. Information-Centric Networking (ICN) is an emerging network architecture that is efficient in content distribution by supporting multicast and in-network caching. Although ubiquitous in-network caching provides nearby retransmission, the design of cache strategy greatly affects the performance of loss recovery. Therefore, how to recover losses efficiently and quickly is an urgent problem to be solved in ICN reliable multicast. In this paper, we first propose an overview architecture of ICN-based reliable multicast and formulate a problem using recovery delay as the optimization target. Based on the architecture, we present a Congestion-Aware Probabilistic Cache (CAPC) strategy to reduce recovery delay by caching recently transmitted chunks during multicast transmission. Then, we propose NACK feedback aggregation and recovery isolation scheme to decrease recovery overhead. Finally, experimental results show that our proposal can achieve fully reliable multicast and outperforms other approaches in recovery delay, cache hit ratio, transmission completion time, and overhead.

Keywords: reliable multicast; loss recovery; ICN; cache strategy; feedback aggregation; recovery isolation

Citation: Duan, Y.; Ni, H.; Zhu, X. Reliable Multicast Based on Congestion-Aware Cache in ICN. *Electronics* **2021**, *10*, 1579. https://doi.org/10.3390/electronics10131579

Academic Editors: Mario Di Mauro, Fabio Postiglione, Besmir Tola, Luigi De Simone and Roberto Natella

Received: 28 May 2021
Accepted: 28 June 2021
Published: 30 June 2021

1. Introduction

Multicasting is a technology that provides an efficient information dissemination method for internet applications by sending information simultaneously from one or more points to a set of other points [1,2]. Many bandwidth-intensive applications (e.g., IPTV, video conferencing, distance education, online gaming) and emerging services such as the Internet of Things (IoT) and Vehicle Networks apply this technology [3–6].

Some packets may be lost when crossing the links between multicast source and receivers. However, conventional multicast systems using the Internet only follow best-effort services [7] and cannot ensure that all multicast data is delivered to each group member reliably and orderly, which may result in a serious performance decline of the application. However, many multicast applications (e.g., critical IoT data delivery such as disaster alarms and industrial internet control commands, and distributing operating system patches or antivirus updates over the network) require that the data must be completely delivered [8]. It can be seen that the research on reliable multicast transmission is of great significance and is essential for improving the performance of multicast applications.

In traditional IP reliable multicast, loss recovery is generally implemented based on retransmission, in which the multicast source simply retransmits the packets once packet losses occur [9]. This leads to great recovery delay. Additionally, as the number of receivers increases, a large number of retransmission requests may overload the sender and the

network, and lead to the well-known negative acknowledgement (NACK) implosion problem. Another problem in retransmission-based reliable multicast is recovery redundancy [10,11], which is mainly caused by recovery exposure and packet repeat.

In recent years, information-centric network (ICN) [12–14] has attracted wide attention. Unlike host-to-host communication in traditional IP networks, ICN focuses on the content object itself rather than the location. ICN names the information by separating the identity and location and uses the corresponding name to replace the IP address as the identifier in network transmission [15]. By naming content in the network layer, ICN supports in-network caching, multicast, and mobility mechanisms to deliver content to users in a more efficient and timely manner [13]. There are two types of content discovery mechanisms, namely lookup-by-name and routing-by-name. Although most ICN architectures provide solutions for multicast [13], they have not addressed the issue of designing efficient and reliable assurance schemes for multicast, even though a great deal of work has been done in the area of IP reliable multicast.

In ICN, users can directly obtain the requested data from the cache node, thus speeding up the content distribution [8]. Moreover, caching mitigates the impact of congestion losses since retransmitted requests can be satisfied by cached data packets [16]. In the same way, when congestion losses occur in multicast, caching provides the possibility for the nearby retransmission. Therefore, multicast can also benefit from caching [16]. Unfortunately, most of the existing work designs cache decision or replacement strategies in ICN from a network-centric performance perspective [17]. Their main purpose is to reduce network traffic by improving cache hit ratio and/or reducing the number of hops. Such methods do not necessarily improve user experience such as the loss recovery delay and content download time in multicast applications requiring reliable delivery. Therefore, designing cache strategies for loss recovery to optimize recovery performance is still a key issue for ICN reliable multicast.

In this paper, to solve the aforementioned problems faced by multicast in ICN, we focus on designing an efficient and reliable multicast approach to improve loss recovery performance. The main contributions of this paper are as follows：

- We introduce a completely reliable multicast architecture in ICN, including four key issues, multicast tree establishment, original multicast data transmission, feedback aggregation, and recovery isolation scheme. Multicast tree node (MTN) responses retransmission requests at any time instead of waiting for the content distribution completion, since storing recently transmitted chunks during original data transmission.
- Based on the above constructed multicast tree, we aim to optimize the normalized loss recovery delay under the constraint of limited cache size and give an in-depth analysis for the proposed optimization model. We propose a distributed Congestion-Aware Probability Cache (CAPC) strategy, in which each MTN makes cache decision individually with certain probability determined by both congestion cost and the cache location on multicast tree.
- To avoid NACK implosion and reduce recovery redundancy, we propose a NACK feedback aggregation scheme and a recovery isolation scheme, respectively. A NACK Table is introduced in the MTN. MTN performs NACK feedback aggregation to ensure that only one NACK per chunk is sent upstream. In addition, MTN controls the propagation range of the recovery data by looking up NACK Table.
- We develop and implement the proposed reliable multicast approach in NS-2 [18], and compare it with the existing approaches and the classic reliable multicast protocol PGM (Pragmatic General Multicast). The experimental results show that the proposal in this paper is superior to other approaches in terms of normalized loss recovery delay, cache hit ratio, transmission completion time, and overhead.

The structure of this paper is organized as follows. Section 2 discusses the related work on reliable multicast. In Section 3, we present the overview of reliable multicast architecture in ICN and make problem formulation using normalized loss recovery delay as the optimization target. In Section 4, we propose the details of Congestion-Aware Probabilistic Cache (CAPC) strategy. Then, we describe the NACK feedback aggregation scheme and recovery isolation scheme in Section 5. Section 6 presents the simulation experiments results and discussion. Finally, we conclude the paper and discuss our plans for the future work in Section 7.

2. Related Work

2.1. Reliable Multicast in IP and ICN

Reliable multicast (RM) means that all multicast data is eventually delivered correctly to each receiver. The common reliability assurance technology is ARQ, FEC, HARQ, network coding, etc. Retransmission is key for the ARQ-based recovery scheme to achieve RM. In this paper, we focus on a retransmission-based RM scheme, which provides a simple and robust solution to ensure that each multicast member receives all multicast packets [9]. Retransmission ways are usually based on a unicast [19], multicast [20], and unicast-multicast hybrid [21].

The feedback implosion problem easily occurs when there are many receivers and poor link quality, which limits the scalability of reliable multicast protocols. There are three main approaches to solve the problem. First, tree-based feedback [22–25] forms a tree-based hierarchical structure for sources and receivers, which can prevent receivers from directly contacting the source to maintain the scalability of large receiver sets. RMTP is a representative work in this type of approach, which is a receiver-driven reliable transmission scheme for non-real-time multicast content delivery [24]. In order to avoid the acknowledgment (ACK) explosion, a group of receivers called designated receivers (DRs) aggregate the ACK status in the local area network and forward it upstream to a higher-level DR or source. When the number of receivers with packet losses exceeds a certain threshold, retransmission is in the form of multicast. Second, router-assisted feedback [26,27] uses special routers to aggregate NACK messages. For instance, Cisco has proposed a classic RM solution called Pragmatic General Multicast (PGM) [26], in which a hierarchy of routers supporting PGM (called network elements (NEs)) is deployed throughout the multicast tree to aggregate feedback from receivers to source. Receivers wait for a time randomly chosen from an interval before unicasting a NACK message to the nearest upstream router, which in turn responds with multicast NACK confirmation (NCF). The process is repeated in a hop-by-hop manner until the source receives the NACK in a reliable manner, and the random delay along with suppression is intended to prevent implosion [26]. However, in practical application, too much traffic is incurred due to multicast NCFs. Third, feedback suppression [20,28]. The receivers wait a random time before multicasting retransmission requests to the entire group. If a receiver receives the same retransmission request from the others, it will refuse to send this request.

Bit-Indexed Explicit Replication (BIER) [29] is a new multicast protocol that removes the need for flow-state in intermediate routers, with each destination explicitly indicated by the source. Intermediate routers replicate and forward packets over the interfaces providing shortest paths (according to unicast routes) to the specified destinations. The authors proposed BIER-based reliable multicast [30] to efficiently retransmit missing packets to the requesting destinations. Source collects NACKs for a certain lost packet for a small amount of time and records the destinations requesting retransmission. When that time expired, source uses BIER to send the retransmission to exactly the set of destinations that have sent the NACK. However, it relies on source to retransmit the recovery packets. To solve this problem, the reliable BIER mechanism was extended to support recovery

from peers in [31]. Rather than being directly sent to the source, NACKs can be first transmitted through an ordered set of peers, each of which may provide retransmission if they have a cached copy of the lost packet.

Several packet loss recovery solutions (e.g., [32,33]) combined multicast technologies with other enabling technologies to enhance their reliability. For instance, Zhang et al. [32] presented an OpenFlow enabled elastic loss recovery solution, called ECast, which acquires packet loss state according to tree-based NACK feedback and calculates packet retransmission method based on elastic area multicast (EAM). It can reduce the recovery redundancy because it does not use irrelevant communication links and does not produce duplicate recovery packets. Mahajan et al. [33] designed and implemented a platform called ATHENA to implement multicast in SDN-based data centers, providing high reliability and congestion control mechanisms to ensure fairness.

Furthermore, several reliable multicast schemes have been proposed for ICN. In [34], proposed is a retransmission-based Reliable Multicast Transport for the Publish/Subscribe Internet (PSI) architecture (RMTPSI) by mapping the ideas of PGM to PSI architecture. It uses selected routers on the multicast tree (as opposed to DR in RMTP) to aggregate feedback and control the propagation of retransmissions. However, the recovery phase and the original content distribution phase are performed separately, and the former must be executed after completion of the latter, resulting in an increase in content distribution completion time. A lightweight enhancement to Content-Oriented Publish/Subscribe System (R-COPSS) was proposed for flow and congestion control as well as for reliability [35]. It adjusts sending rate to accommodate the ACKer that is selected according to the loss rate and throughput periodically fed back by all the subscribers. The slower subscribers obtain the lost packets via local repair. Moreover, [36] proposed a Network Coding-based Real-time Data Retransmission (NC-RDR) algorithm for ICN multicast to dynamically combine the missing packets by using random linear coding.

In addition, with regard to the unreliability of data transmission or service caused by network failures, some strategies related to the redundancy of network components (e.g., virtual machines, router, etc.) [37–39] have been proposed to protect network components from network failures.

2.2. Cache Strategy in Reliable Multicast

Several cache strategies have been proposed in IP reliable multicast. Active reliable multicast (ARM) is a new loss recovery scheme for large-scale reliable multicast, which emphasizes the active role of the router [40]. The active routers follow configuring hierarchical multicast tree, and support caching, NACK consolidation, and scoped retransmission. However, ARM did not consider cache utilization efficiency and had difficulty caching the packets efficiently with limited cache size. In [41], the authors studied and compared the combinations of three cache policies, namely, the timer-based, simple FIFO (S-FIFO), and probabilistic FIFO (P-FIFO); and three cache allocation schemes, equal sharing, least requirement first (LRF), and proportional allocation, and found that P-FIFO with proportional cache allocation performs the best in most cases. Xie et al. [42] formulated the cache policy design as an optimization problem and proposed an algorithm called Optimal Caching Time (OCT) for determining the caching lifetime of packets. These caching strategies are based on packets. In [43], the authors proposed a Network Coding-based FIFO (NCFIFO) caching policy to extend the caching lifetime of the packets at the recovery nodes without dropping any of the incoming packets. But it increased the complexity of router operation.

ICN enables rapid data retrieval due to its native in-network caching mechanism, thus shares properties with cache-based reliable multicast protocols, by enabling data recovery from nearby routers [31]. In [44], the authors found that using ICN in-network packet-level cache for retransmission can reduce the expected retransmission latency and is a valuable error control method. However, they did not study the impact of different cache strategies on error recovery in unicast or multicast scenarios.

Finally, as shown in Table 1, we present a summary of the reliable multicast approaches in the literature.

Table 1. Summary of the reliable multicast approaches in the literature.

Related Work	Scenario	Feedback Aggregation/Suppression Scheme	Loss Recovery Method
RMTP [24]	IP	Tree-based feedback, using DRs to aggregate and forward ACK upstream to a higher-level DR or source	Multicast source retransmission
PGM [26]	IP	Router-assisted feedback, deploying NEs to aggregate NACK and multicast NCFs downstream to suppress NACK	Multicast source retransmission
BIER-based RM [30]	IP	Multicast source aggregating NACK from receivers in a certain period of time	Multicast source retransmission with BIER scheme
NCFIFO [43]	IP	-	Using Network Coding-based FIFO cache policy to extend the lifetime of the cached packets, recovery node retransmission
RMTPSI [34]	ICN	Using selected routers on the multicast tree (as opposed to DRs in RMTP) to aggregate feedback and control the propagation of retransmissions	Mapping PGM to PSI architecture, source retransmission, and recovery phase must be executed after completion of original content distribution
R-COPSS [35]	ICN	Rendezvous Point (RP)-based FIB propagation (towards publisher), using a PIT entry pointing to the RP direction for all subscribers in the subtree, and local FIB propagation	Local repair, subscribers selectively requesting repair from others based on application needs
NC-RDR [36]	ICN	-	Using random linear coding to dynamically combine the missing packets, sink routing node retransmission
CAPC-based RM proposed in this paper	ICN	Based on Chunk-level NACK, bitmap, and the NACK Table with Timer to aggregate NACK feedback	Congestion-aware probabilistic cache, MTNs or source retransmission, recovery isolation by using NACK Table

3. Overall Architecture

In this section, we first introduce the overview of reliable multicast architecture in ICN and present its pivotal components including the establishment of multicast tree, original multicast data transmission, NACK feedback aggregation, and recovery isolation in Section 3.1. Then in Section 3.2, we build a network model using the normalized loss recovery delay as the optimization target, and formulate the problem of content placement during original data transmission as the congestion cost saving maximization problem and perform some analysis.

3.1. Architecture Overview

Some ICN solutions propose to deploy clean-slate ICN, but the high cost limits their deployment, such as Publish Subscribe Internet Technology (PURSUIT) [45], Named Data Networking (NDN) [16], and Content Centric Networking (CCN) [46]. In order to smoothly evolve, the ICN scheme in our reliable multicast approach is similar to MobilityFirst [47], which can coexist with existing IP networks to achieve incremental deployments. And the content discovery mechanism uses look-up-by-name instead of routing-by-name represented by CCN [46] and NDN [16]. According to ICN's main principle of separating the identity and location, in the network, the elements such as contents, devices, and services can be regarded as entities. Each entity is assigned an Entity-ID (EID) as the identifier (or name), and the Network Address (NA) is used as the locator. We adopt IP address as NA to be compatible with existing IP-based networks. As a result, the identifier is completely separated from the locator, and they are dynamically bound together

through Name Resolution System (NRS). Routers can forward ordinary IP packets according to IP address and ICN packets according to EID due to maintaining both IP routing information and name forwarding information. Like most ICN architecture, the group of packets, so called as chunk, are the basic cache unit in our architecture. Moreover, the content object is split into chunks, of which all packets are transmitted along the multicast tree.

As an entity, each multicast service is assigned with a Globally Unique Multicast Service Identifier (GUMSID). Since NRS maintains the mapping between the GUMSID and multiple MTNs NAs, regardless of the location of the multicast source, multicast receivers can easily join or leave the multicast service based on the GUMSID. Every MTN maintains the multicast name forwarding table (MNFT), which is composed of a set of multicast name forwarding entries (MNFEs). Each MNFE includes an in-interface, GUMSID, and out-interface list. As shown in Figure 1a, we consider a multicast system consisting of multicast sources, routers, Name Resolution System (NRS), and multicast receivers.

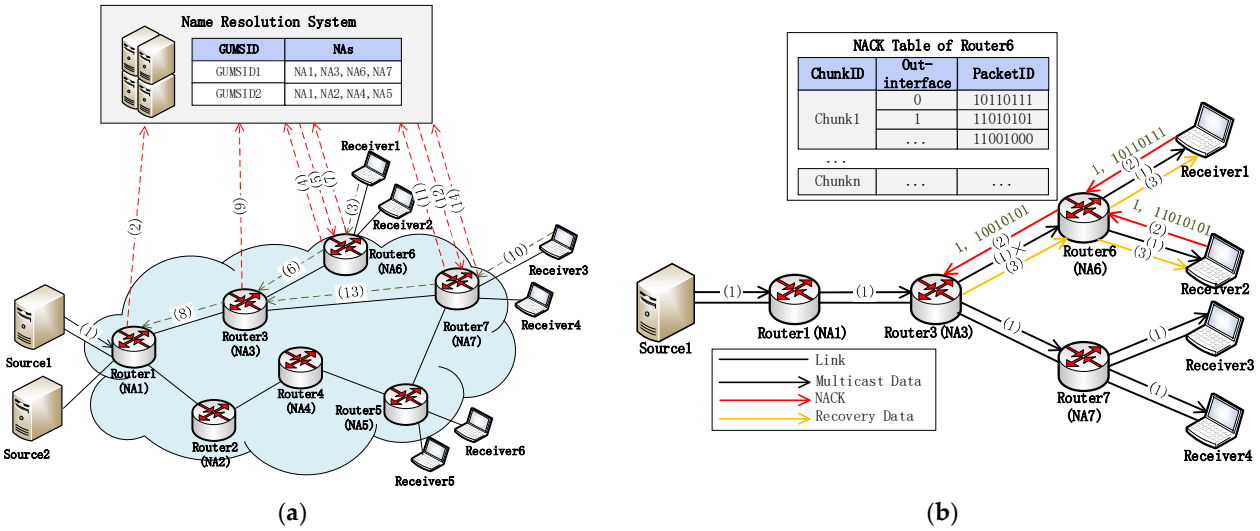


Figure 1. Overview of reliable multicast architecture in ICN. (**a**) The establishment process of multicast tree; (**b**) multicast data transmission and loss recovery process.

3.1.1. The Establishment of Multicast Tree

Here, we take GUMSID1 in Figure 1a as an example to analyze the establishment of multicast tree. Firstly, multicast source1 randomly selects a router (e.g., Router1) to send multicast data (arrow 1). Then Router1 initializes the MNFE and registers the mapping between GUMSID1 and NA1 with NRS (arrow 2), and serves as the root node of multicast tree. When multicast Receiver1 is interested in multicast service GUMSID1, it sends the join message containing the GUMSID1 (arrow 3). After receiving join message, Router6 sends resolution request to NRS to obtain the MTNs' NAs of the GUMSID1 (arrow 4). At this time, only NA1 is returned (arrow 5). Then the join message is sent to NA1 hop by hop. The specific process is as follows. Router6 first sends join message to Router3 (arrow 6). Then Router6 initializes the MNFE and registers the mapping between GUMSID1 and NA6 with the NRS (arrow 7). After receiving the join message, Router3 performs the same operations (arrow 8-9) as the operations indicated by arrow 6-7. Then, Router1 updates MNFE of GUMSID1 and sends join-ack message along the reverse path of the join message. At this point, the reverse path of the join message becomes a branch of the multicast tree. When Receiver3 sends a join message to join multicast service GUMSID1 (arrow 10), Router7 sends resolution request to NRS (arrow 11) and obtains that the MTNs' NAs corresponding to GUMSID1 are NA1, NA3, and NA6 (arrow 12). Router7 randomly selects an MTN NA (e.g., NA3) to join GUMSID1 and sends join message to NA3 (arrow 13).

Meanwhile, Router7 initializes the MNFE and registers the mapping between GUMSID1 and NA7 with the NRS (arrow14). Then, Router3 updates MNFE of GUMSID1 and sends join-ack message along the reverse path of the join message. So far, the multicast tree construction of GUMSID1 has been completed. Note that the selection mechanisms of root node and MTN involved in the above process are not the focus of our study, so we simply use the random selection method. Moreover, in terms of maintaining the multicast group membership between the multicast receiver and its directly connected MTN, the multicast receiver identifies a multicast service through the GUMSID rather than IP multicast address in IGMP [48] and MLD [49].

3.1.2. The Transmission of Multicast Data

The multicast tree of GUMSID1 has been constructed, as shown in Figure 1b. Next, we focus on the multicast data transmission and how to ensure its reliability.

Step 1: The MTN forwards original multicast data packets (ODATAs) according to the stored MNFT. After receiving a multicast packet, the MTN lookups MNFT according to the incoming interface and the GUMSID carried in the packet. If there is a matching MNFE, the multicast packet is cloned and forwarded out from the corresponding out-interface respectively, otherwise, the multicast packet is discarded. In addition, in the process of ODATAs transmission (arrow 1), the MTNs perform caching for ODATAs in chunk that is a group of packets. Therefore, the MTNs can handle retransmissions of lost packets by only storing recently transmitted chunks. How to design cache strategy is a critical issue, which directly affects the performance of loss recovery. To decrease the loss recovery delay, this paper proposes a Congestion-Aware Probability Cache strategy, which is described in detail in Section 4.

Step 2: Assuming that ODATAs losses occur on the link between Router3 and Router6 in Figure 1b during multicast data transmission. Receiver1 and Receiver2 detect packet loss by inspecting the sequence number, then use Chunk-level NACK and bitmap to aggregate and feedback packet loss information of a chunk (arrow 2). In principle, no matter how many packets are lost in a chunk, only one NACK is consumed. The MTN maintains a NACK Table and re-aggregates the NACK of the same chunk to ensure that only one NACK for a chunk is sent to the upstream.

Step 3: When NACKs reach an MTN that caches the complete chunk, recovery data packets (RDATAs) will be forwarded to the downstream immediately, otherwise after NACK aggregation, NACKs continue to be forwarded upstream until the multicast source. In Figure 1b, Router3 happens to cache the complete chunk, so it directly retransmits RDATAs to Receiver1 and Receiver2. In the recovery phase (arrow 3), the MTNs also control the propagation of RDATAs. They look for NACK Table before forwarding RDATAs. Then RDATAs are only forwarded to the downstream MTNs or receivers that have sent NACKs of the chunk, so that recovery isolation can be achieved. The proposed feedback aggregation scheme and recovery isolation scheme are detailed in Section 5.

3.2. Problem Statement

Under the constructed multicast tree, our objective is to optimize the loss recovery delay of reliable multicast. Therefore, in this section, we establish a network model and set up the problem formulation for the purpose of loss recovery in reliable multicast, and use recovery delay as the optimization target. Table 2 represents the main notations used in this paper.

Table 2. Summary of the notations.

Notation	Description
R	The number of multicast receivers
V	Set of MTNs
L	Set of lost chunks
M_r	The number of chunks experiencing loss for receiver r
T^j	The recovery time for chunk j from multicast source to receiver r (including retransmission request)
T_i^j	The RTT between router i and multicast source when the router i hit by NACK for chunk j
K	Set of all chunks
b^j	The size of the chunk j
C_i	The cache capacity of the router i
q_i^j	The congestion cost of chunk j observed at router i
h_i^j	The hop count from router i to multicast source or the nearest upstream cache router on multicast tree for chunk j
$d_{r,root}$	The hop count from receiver r to the root node of multicast tree

3.2.1. Model Description

We introduce a three-layer multicast tree structure to set up and illustrate our model, and the multicast tree topology of any number of layers still applies to the model. As shown in Figure 2, assuming that a multicast tree has been set up and a fixed amount of cache has been allocated to each router along the multicast tree. The multicast tree consists of R receivers, V MTNs, and a multicast source. For ICN multicast, the advantage of caching is that the data packet can be retransmitted through the cache node without having to request retransmission from the original multicast source far away. The cache strategy specifies which chunk should be cached at router and how long it should be cached. A good cache strategy should minimize the recovery delay.

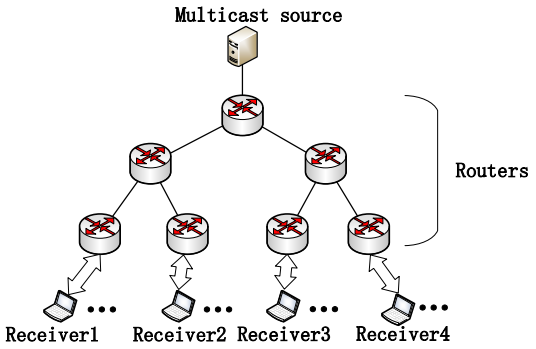


Figure 2. A three-layer multicast tree model.

3.2.2. Problem Formulation

This section formulates the average loss recovery delay for all receivers in reliable multicast. First, the average loss recovery delay $\bar{D}_r$ for receiver r is defined as

$$\bar{D}_r = \left(\sum_{j}^{L} T^j - \sum_{j}^{L}\sum_{i}^{V} Z_i^j T_i^j\right) \Big/ M_r \quad (1)$$

Let $\overline{RTT}$ be the mean round trip time from the multicast source to all receivers in the given multicast group. Then the design goal of cache strategy in reliable multicast is to minimize the normalized loss recovery delay $\bar{D}$, which can be expressed using the following equation:

Minimize:

$$\overline{D} = \frac{\frac{1}{R}\sum_{r=1}^{R}\overline{D}_r}{\overline{RTT}} \tag{2}$$

Subject to:

$$z_i^j = \{0/1\} \tag{3}$$

$$\sum_{j\in K} b^j z_i^j \le C_i, \quad \forall i \in V \tag{4}$$

where z_i^j be an integer variable in $\{0,1\}$, which specifies whether node i caches chunk j ($z_i^j = 1$) or not ($z_i^j = 0$).

The first term in the numerator of Equation (1) represents the total loss recovery delay without cache for receiver r, and the second term in that represents the total RTT between the nodes hit by NACKs and multicast source. Since only the second term is related to the caching position of chunks, the maximization of the second term is equivalent to the minimization problem of Equation (2). It means that caching location on the multicast tree as a factor should be considered in the cache strategy for the decrease in loss recovery delay.

Moreover, the main cause of packet loss on wired links is congestion, and the main manifestation of congestion is packet loss [50]. The more serious the congestion condition is, the greater the packet loss probability is. Therefore, congestion condition is the other important factor for cache strategy design in ICN reliable multicast. In the event of congestion loss, the recovery of lost packets generally relies on retransmission, which causes recovery delay overhead. From this perspective, if we regard recovery delay caused by congestion loss as a cost (collectively referred to as congestion cost in the paper), an approximation of the normalized loss recovery delay optimization problem in Equations (1)–(4) is to maximize the hop count weighted congestion cost saving because of caching.

Maximize:

$$\sum_{i\in V}\sum_{j\in K} q_i^j \frac{h_i^j}{d_{r,root}} z_i^j \tag{5}$$

Subject to:

$$z_i^j = \{0/1\} \tag{6}$$

$$\sum_{j\in K} b^j z_i^j \le C_i, \quad \forall i \in V \tag{7}$$

$$0 < h_i^j \le d_{r,root}, \quad \forall i \in V \tag{8}$$

where q_i^j is the congestion cost of chunk j observed at node i. b^j is the chunk size of chunk j, and C_i is the cache capacity of router i. $\frac{h_i^j}{d_{r,root}}$ is a weighting factor related to hop count from the node i to multicast source or the nearest upstream cache node on the multicast tree for chunk j.

A straightforward heuristic common in cache for this integer programming problem is to let MTNs cache chunks with highest congestion cost. However, if the MTNs cache the chunks with the highest cache congestion cost and purge the chunk with the lowest congestion cost when the cache space is exhausted, all the cache space will be filled with the chunks with the highest congestion cost after a period of stability. Ultimately, the lifetime of chunks with high congestion cost is too large, so that some chunks with lower congestion cost experiencing loss cannot benefit from caching, resulting in failure to improve the recovery delay performance. Therefore, we propose a congestion-aware probabilistic cache strategy to partially cache the chunks with relatively high congestion cost, which is described in detail in the next section.

4. Congestion-Aware Probabilistic Cache Strategy

The core mechanism of congestion-aware caching is the congestion evaluation for each passing content object [51]. In a multicast environment, it is impractical for receivers to feedback the congestion condition of each link and node through which the chunk passes. In fact, when MTNs have collected all the congestion conditions, the recovery stage may already be in progress. Therefore, making cache decision at this time is of little significance in reducing the recovery delay. Considering the timeliness of acquiring parameters and simplifying the operation of routers in practice, we present a distributed probabilistic cache strategy to reduce loss recovery delay in reliable multicast, called CAPC (Congestion-Aware Probabilistic Cache), in which each MTN makes cache decision individually with local information. Note that the FIFO replacement strategy is used to keep the newly transmitted chunks in the cache when the cache space is insufficient.

4.1. An Overview of CAPC

When receiving a complete chunk, each MTN decides whether to cache it with a certain probability, which is related to congestion cost and cache location on the multicast tree.

The congestion cost reflects the congestion condition that a chunk experiences in a node, which is related to the probability of packet loss. Moreover, considering the influence of the location of the cache node on the multicast tree, we also introduce a weighting factor related to the hop count to weight. Therefore, the formula of cache probability p_i^j for chunk j in node i is calculated as

$$p_i^j = q_i^j \times \frac{h_i^j}{d_{r,root}} \tag{9}$$

where, q_i^j reflects the congestion cost of chunk j in the node i, whose specific definition is introduced in Section 4.2. $\frac{h_i^j}{d_{r,root}}$ means the relative weighted factor and its value is between 0 and 1.

Additionally, the *mark* field is added to the header of ODATA to characterize whether it is the last packet of the chunk. For the last packet of a chunk, the multicast source sets *mark* field to 1. If the chunk is received completely, under the CAPC, each MTN simply makes a decision to cache it or not according to the cache probability calculated by Equation (9). If the cache probability is greater than or equal to the cache probability threshold P_{th}, the MTN will cache it. The detailed cache probability calculation process is presented in Algorithm 1.

Algorithm 1 The Calculation of Cache Probability

Input: $q_{len,i}$, Q_{max}, h_i^j, $d_{r,root}$, ODATA
Output: p_i^j

initialization: $\mu = 0.9$, $Q_{low} = 0.25 * Q_{max}$, $Q_{high} = 0.75 * Q_{max}$
for each incoming packet (ODATA) with a chunk j in node i **do**
  if *mark* == 0 **then**
    update the current weighted moving average queue length q_{av} by Equation (10)
  else if *mark* == 1 **then**
    if $q_{av} < Q_{low}$
      $q_i^j = 0$
    else if $Q_{low} \le q_{av} \le Q_{high}$
      calculate congestion cost q_i^j of chunk j in node i by Equation (12)
    else
      $q_i^j = 1$

12: **end if**
13: $p_i^j = q_i^j \times \frac{h_i^j}{d_{r,root}}$
14: **end if**
15: **end for**

4.2. Congestion Cost for Cache Management

Inspired by classic RED algorithm [52], CAPC takes average queue length as the indicator to evaluate the congestion condition of the MTN through which a chunk passes, and perceives congestion cost by detecting the average queue length. In some RED algorithms, the packet loss rate has a linear relationship with the average queue length. The congestion cost calculated by those algorithms is particularly small. In order to cache more chunks between the threshold Q_{low} and Q_{high}, this paper proposes an improved algorithm to calculate the congestion cost.

4.2.1. Weighted Moving Average of Queue Length

When receiving i-th packet of chunk j, the MTN acquires the current queue length $q_{len,i}$. Then, we estimate and smooth queue length using EWMA calculation model [53] as current average queue length as follows:

$$q_{av,i} = \mu \times q_{len,i} + (1 - \mu) \times q_{av,i-1} \tag{10}$$

where μ $(0 < \mu < 1)$ is a weighting factor.

4.2.2. The Calculation of Congestion Cost

After receiving the last packet of the chunk, the result calculated in Equation (10) is as the average queue length of the chunk j. Then, we compute the value of q_i^j by comparing the average queue length with two threshold values of router queue length, named Q_{low} and Q_{high}. The value of q_i^j is determined as

$$q_i^j = \begin{cases} 0, & q_{av} < Q_{low} \\ 1 * \sqrt{(Q^2{}_{high} - Q^2{}_{low}) * q_{av} + Q^2{}_{low}}), & Q_{low} \le q_{av} \le Q_{high} \\ 1, & q_{av} > Q_{high} \end{cases} \tag{11}$$

If the average queue length is less than Q_{low}, the network load is considered to be relatively light. In this case, the probability of the chunk experiencing packet loss is very low and it is not worth caching. Therefore, the congestion cost is set to 0. When average queue length is larger than Q_{high}, we consider the current congestion condition as heavy at the MTN. At this point, the chunk experiences packet loss with a high probability, thus congestion cost is determined as 1. Between the threshold Q_{low} and Q_{high}, congestion cost increases as the average queue length increases.

$$q_i^j = \begin{cases} 0, & q_{av} < Q_{low} \\ 1 * \dfrac{\sqrt{(Q^2{}_{high} - Q^2{}_{low}) * q_{av} + Q^2{}_{low}}) - \sqrt{(Q^2{}_{high} - Q^2{}_{low}) * Q_{low} + Q^2{}_{low})}}{\sqrt{(Q^2{}_{high} - Q^2{}_{low}) * Q_{high} + Q^2{}_{low}}) - \sqrt{(Q^2{}_{high} - Q^2{}_{low}) * Q_{low} + Q^2{}_{low})}}, & Q_{low} \le q_{av} \le Q_{high} \\ 1, & q_{av} > Q_{high} \end{cases} \tag{12}$$

Finally, in Equation (12), we use Min-Max scaling to normalize the q_i^j between the threshold Q_{low} and Q_{high} to the range of [0,1]. As shown in Figure 3, when the average queue length is between the threshold Q_{low} and Q_{high}, our proposed algorithm can achieve greater congestion cost than the traditional RED algorithm.

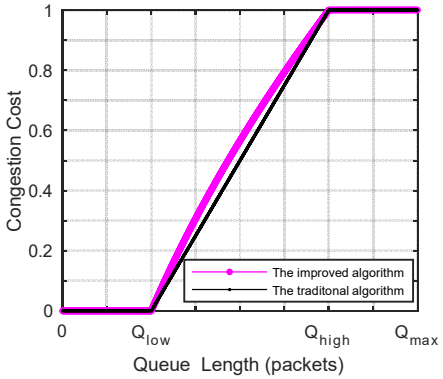


Figure 3. Relationship between congestion cost and queue length.

4.3. The Processing of Hop Count

In this section, we propose relevant algorithms on how to acquire hop count h_i^j and receiver-root distance $d_{r,root}$ in each MTN.

In the process of multicast tree establishment, the join message carries hop count of routers through which it has passed. The hop count increases by one every time the join message passes through a router. The join-ack message carries the total hop count in a receiver-root path and is transmitted to each router on the path in the reverse direction. If a new member joins a branch of the multicast tree, the updated hop count information will be diffused to other MTNs on the receiver-root path after the joining process is completed. Ultimately, each MTN records the hop counts of all receivers-to-root paths through which it passes, and takes the maximum value as $d_{r,root}$.

To acquire h_i^j, we extend the header of the ODATA and add the *hop count* field to carry hop count information. For the last packet of a chunk, the multicast source initializes *hop count* field to 1. As described in Algorithm 2, if the MTN decides not to cache the chunk, the *hop count* field is increased by 1 before forwarding the last packet to the downstream, otherwise, the *hop count* field is reset to 1.

Intuitively, under the same congestion cost, if a chunk has not been cached at the upstream of the MTN, then h_i^j is larger, therefore it is more likely to be cached. Conversely, if the chunk has been cached at a certain upstream MTN before it reaches the MTN, there is a decrease in the cache probability. Besides weighing congestion cost saving, the other advantage of using *hop count* is to avoid excessive duplication of cached chunks in the network.

Algorithm 2 ***Hop Count*** **Field Processing in the MTN**

```
for each incoming ODATA of a chunk j do
    if mark == 1 then
        if cache chunk j then
            hop count field of the ODATA header = 1
        else if
            hop count field of the ODATA header + 1
        end if
        forward the ODATA via the corresponding out-interface
    end if
end for
```

5. NACK Feedback Aggregation and Recovery Isolation Scheme

5.1. NACK Feedback Aggregation Scheme

For the sake of reducing the additional traffic for loss recovery, NACKs are sent by receivers in chunks instead of packets in our proposal. A NACK carries all packet loss information for a chunk, consisting of a chunk number (ChunkID) and a bitmap. For instance, a NACK with a ChunkID of 20 and a bitmap of 10111010. The zeros in the second, sixth and eighth bits indicate that packet with packet sequence number (PacketID) 1, 5, and 7 for chunk 20 have been lost.

Moreover, this paper introduces the NACK Table as shown in Figure 1b. In the example of GUMSID1 in Figure 1b, each MTN maintains a NACK Table to record the NACK information of GUMSID1, including ChunkID, the interfaces that have received the NACK, and bitmap information carried by NACK. In MTN, NACK is aggregated by means of the timer T_{agg}. Additionlly, NCF packets in the PGM are no longer used in our scheme. When an MTN receives the first NACK (e.g., 1, 10110111) of Chunk1, it does not immediately forward it to the upstream MTN but generates NACK entry for the chunk and starts an aggregation timer with T_{agg}. During the waiting period of T_{agg}, if the MTN receives NACKs (e.g., 1, 11010101) from other downstream MTNs, the NACK entry of the chunk is updated. Upon the aggregation timer expires, the MTN aggregates all the loss information of downstream MTNs by bitmap bitwise logical AND operation, then generates and sends a NACK (e.g., 1, 10010101) to the upstream MTN immediately. This scheme can ensure that each receiver and each MTN only send one NACK to the upstream MTN for a chunk. There is no doubt that the scheme can suppress NACK and avoid NACK storms.

In addition, each ChunkID entry of NACK Table has a life cycle. For a specific ChunkID, the life timer with T_{live} should be reset whenever a new NACK is received. If no new NACK is received during T_{live}, it is considered that the losses have been completely recovered. Then, the NACK entry will be deleted. The setting of T_{live} enables the entry of the restored chunk to be deleted in time, so that the NACK table does not occupy too much storage space.

When MTN i receives a NACK, if cache hits, it forwards RDATAs to the corresponding downstream MTNs immediately. Otherwise, it performs NACK feedback aggregation according to Algorithm 3.

Algorithm 3 NACK Feedback Aggregation Scheme

```
for each incoming NACK packet for chunk j in MTN i do
    if the life timer with T_live for chunk j has not expired then
        reset T_live
        if NACK entry for chunk j exists then
            if the aggregation timer with T_agg for chunk j has not expired then
                update the NACK entry for chunk j
            else
                perform bitmap bitwise logical AND operation for chunk j
                generate a new NACK packet and send it to the upstream MTN
            end if
        else if NACK entry for chunk j does not exist then
            generate the NACK entry for chunk j
            set T_live
        end if
    else if life timer with T_live for chunk j has expired then
        delete the NACK entry for chunk j and drop the NACK packet
    end if
end for
```

5.2. Recovery Isolation Scheme

Recovery isolation refers to the attribute that recovery data packets are only sent to the local area where the data packet is lost. Ideally, the receiver receives the recovery packet only if it misses this ODATA. To avoid recovery exposure and recovery packet repeat, this paper proposes a recovery isolation scheme to support the attribute.

As mentioned in the previous section, NACK Table is introduced to record packet loss information for each chunk. As shown in Figure 1b, when RDATA arrives at an MTN, the MTN looks up the NACK entry of the chunk to which the RDATA belongs, then only forwards RDATA towards the downstream MTNs that have sent NACK for the chunk. As a result, those downstream MTNs and receivers that have not forwarded the NACK for the chunk do not receive RDATA.

In actual operation, NACK packets and RDATAs may be lost. To achieve fully reliable multicast, a retry timer with T_{retry} is reset by receivers after the NACK is sent. When the timer expires, if no corresponding RDATAs have been received, receivers resend another NACK of the chunk to upstream MTN until RDATAs are received.

To illustrate the difference between the proposed recovery isolation scheme and other retransmission methods, we still use the example in Figure 1b, assuming that packet losses occur on the link between Router3 and Router6. In Figure 4a, the multicast-based retransmission method greatly wastes bandwidth resource because RDATAs are always retransmitted to the whole group. While with unicast-based retransmission method in Figure 4b, repeat traffic is transmitted on the same link since it independently retransmits RDATAs to the receivers that have sent retransmission requests. Hence, it brings about great recovery redundancy. As represented in Figure 1b, the proposal in this paper does not take up extraneous links and does not generate duplicate RDATAs, thus the recovery isolation scheme can be applied to ensure that the recovery traffic footprint is minimal.

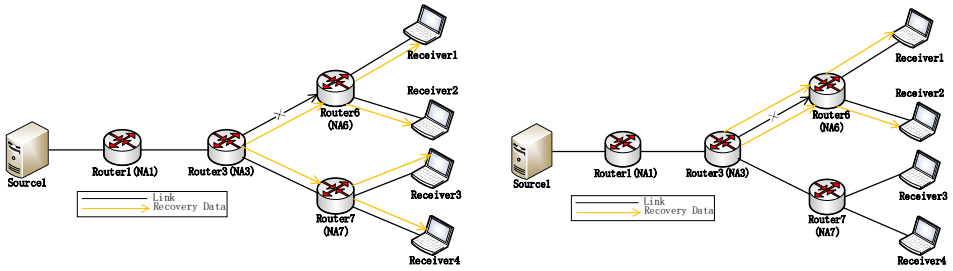


Figure 4. Comparison of different reliable multicast retransmission methods: (**a**) multicast-based retransmission method; (**b**) unicast-based retransmission method.

6. Performance Evaluation

In this section, we conducted a series of experiments to evaluate the performance of the proposed reliable multicast approach. To this end, we developed CAPC, ProbCache [54], Prob [55], leave copy everywhere (LCE), and No-Cache based on the proposed reliable multicast architecture over NS-2 [18], which is a discrete event simulation tool. We used the implementations for PGM [26] that previously existed in NS-2. We compared CAPC with ProbCache, Prob, LCE, and No-Cache based on the proposed reliable multicast architecture and with classic IP reliable multicast protocol PGM at the aspect of loss recovery delay, cache hit ratio, transmission completion time, and overhead.

6.1. Simulation Setup

The established multicast tree structure and the bandwidths of links are shown in Figure 5a. The propagation delay of links was set to 10 ms. We focused on the multicast source1 (GUMSID1) deployed with the reliable multicast approach. The background traffic with multicast source2 (GUMSID2) was introduced just to increase network traffic load, which shared the same multicast tree as GUMSID1. In various experiment scenarios, there were always four receivers for GUMSID2, and each of them was connected to an end MTN. The number of receivers of GUMSID1 was variable. We took 80 receivers for GUMSID1 as an example, and the simulation topology in NS-2 is shown in Figure 5b. The pink, green, and blue circles represent the multicast sources, MTNs, and receivers, respectively. The rates of the two multicast sources were set to 200 Mbps. In order to accurately evaluate and verify the performance of the proposed approach, all the packet losses were caused by congestion in the experimental setting. Each experiment in the same scenario was independently run ten times and the average values were calculated.

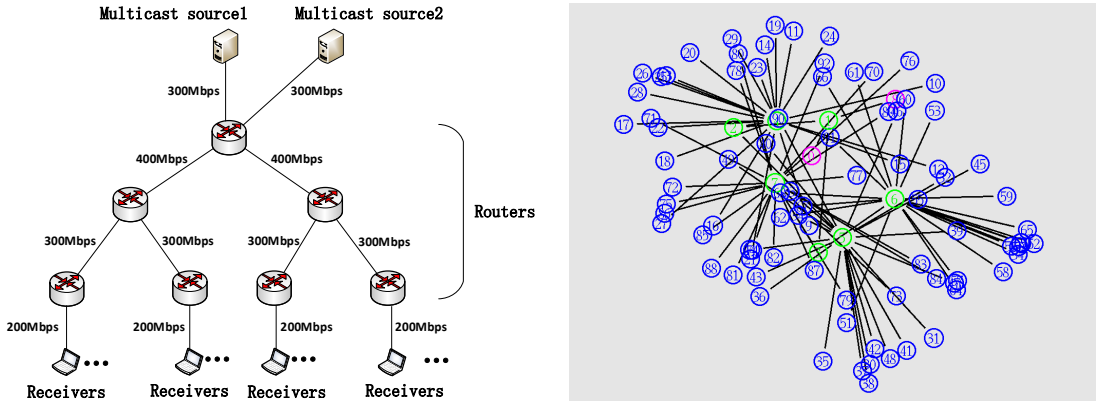


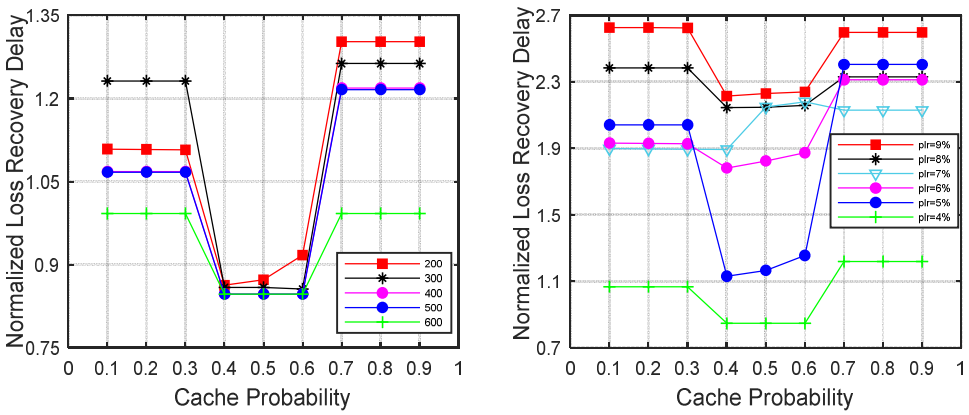
Figure 5. The simulation topology. (**a**) The multicast tree structure and the bandwidths of links in simulation topology. (**b**) The simulation topology with 80 receivers for GUMSID1 and 4 receivers for GUMSID2 in NS-2.

Table 3 shows the basic parameters and their values for our simulation. Moreover, in the initialization phase of Algorithm 1, the values of μ, Q_{low}, and Q_{high} were set according to [56]. In order to obtain the suitable value of P_{th}, we performed a lot of experiments with the number of receivers of 40 under different cache sizes when the packet loss rate was set to about 4% and under different packet loss rates when the cache size in each MTN was set to 400, respectively. As shown in Figure 6, when P_{th} is set to 0.4, the normalized loss recovery delay is the lowest. In our experimental setting, P_{th} is set to 0.4 and 0 for CAPC and LCE, respectively. For Prob, the router decides to cache a chunk at random with a fixed probability that is set to the average cache ratio of CAPC under the same conditions. For ProbCache, Timesin factor was set to five. Additionally, when cache space is exhausted, CAPC, ProbCache, Prob, and LCE use FIFO replacement strategy.

Table 3. Experiment parameters.

Parameter	Value
The maximum queue length (Q_{max})	100–2000 packets
The upper threshold for queue length (Q_{high})	$0.75 \times Q_{max}$
The lower threshold for queue length (Q_{low})	$0.25 \times Q_{max}$
Cache size	100–700
Chunk size	10 packets
Packet size	1032 bytes
The number of receivers for GUMSID1	8,20,32,40,44,56,68,80
Retransmission timer T_{retry}	$1.5 \times RTT$
Aggregation timer T_{agg}	0.023–0.035s
Life timer T_{live}	$4 \times RTT$
Weighting factor μ	0.9
Cache probability threshold P_{th} for CAPC	0.4

In the following experiments, when experiments were performed to compare the effects of cache size on various indicators, the packet loss rate and the number of receivers were set to about 4.5% and 40. In addition, PGM has nothing to do with cache size. When experiments were performed to compare the effects of packet loss rate on various indicators, the cache size of each MTN and the number of receivers were set to about 400 and 40. When experiments were performed to compare the effects of the number of receivers on various indicators, the packet loss rate and the cache size of each MTN were set to about 3.5% and 200.



(**a**) (**b**)

Figure 6. Relationship between normalized loss recovery delay (NLRD) and cache probability. (**a**) Relationship between NLRD and cache probability under different cache sizes; (**b**) relationship between NLRD and cache probability under different packet loss rates.

6.2. Loss Recovery Delay

The loss recovery delay is defined as the time interval between a receiver first detecting the packet loss and the receiver receiving the recovery packet. We evaluate the loss recovery delay by normalized loss recovery delay (NLRD) $\bar{D}$ that is defined in Equation (2).

In Figure 7a–c, the results obtained for NLRD are presented. All of the lost packets are retransmitted by source in PGM and No-Cache. The LCE caches any chunk along the way, which causes frequent cache replacements due to the limited cache space. It results in the cached chunks being easily discarded before the corresponding retransmission requests arrive. The ProbCache and the Prob only cache chunks at random, and some chunks that do not experience loss occupy the cache space, therefore only a small number

of lost packets can be recovered by MTNs. Conversely, CAPC takes the congestion condition and cache location on the multicast tree into account, therefore can provide a better local loss recovery service. Consequently, as shown in Figure 7, CAPC achieves the lower NLRD than ProbCache, Prob, LCE, No-Cache, and PGM at all test scenarios.

From Figure 7a, as the cache size of per node increases, CAPC, ProbCache, Prob, and LCE reduce NLRD because more chunks have been cached. Meanwhile, CAPC keeps its advantage at different cache sizes and it performs on average 23.51, 26.20, and 45.47% lower in NLRD compared with ProbCache, Prob, and LCE, respectively. However, PGM and No-Cache have nothing to do with cache size because they do not support cache. Figure 7b shows the effect of packet loss rate on NLRD. The NLRD increases in all approaches with larger packet loss rate. However, CAPC always maintains the lowest NLRD. Increasing loss rate greatly raises the recovery time of PGM under higher packet loss rate. It means that PGM is badly affected by changes in the network condition. Figure 7c shows the effect of the number of receivers on NLRD. Except for CAPC and PGM, NLRD of other strategies increases when the number of multicast receivers increases. Moreover, NLRD of CAPC is obviously the lowest. Furthermore, NLRD of CAPC operates stably under different number of receivers, demonstrating the excellent scalability of our proposal.

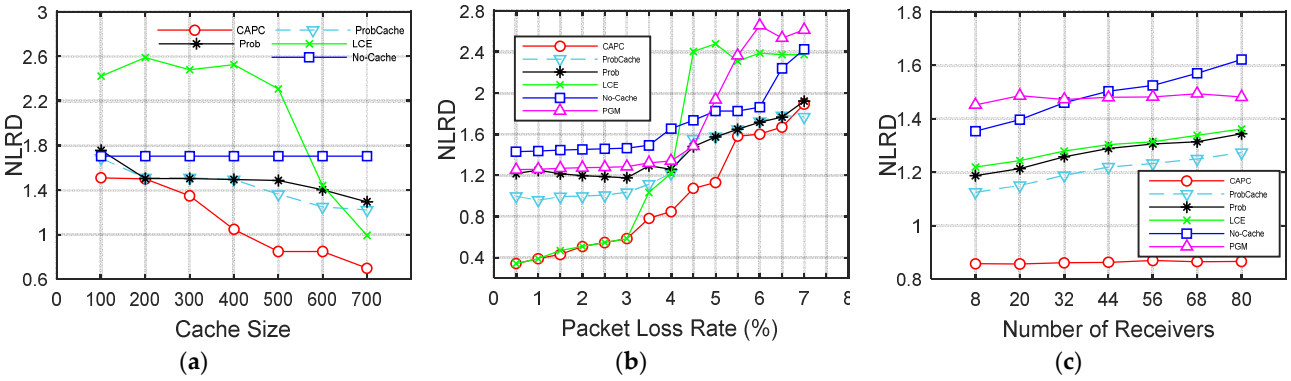


Figure 7. The comparison of normalized loss recovery delay (NLRD). (**a**) The comparison of NLRD under different cache sizes; (**b**) the comparison of NLRD under different packet loss rates; (**c**) the comparison of NLRD under different numbers of receivers.

The above results also support the comment, given in the following subsection, which says that the increase in transmission completion time is mainly due to the increase in loss recovery time.

6.3. Average Cache Hit Ratio

With a cache hit, the cache node retransmits the requested data. The cache hit ratio is defined as the number of NACKs recovered divided by the number of NACKs received in a cache node. Graphics given in Figure 8 illustrate the variation of average hit ratios of all MTNs against different network parameters. Note that PGM and No-Cache do not participate in the comparison of this indicator. Because they do not support cache, the cache hit ratio is always 0.

From Figure 8a, cache hit ratios of CAPC and LCE become greater with the increasing of cache size. This is an expected result of caching more chunks in greater cache space. Furthermore, CAPC keeps the best performance over other strategies excluding the cache size of 100. From Figure 8b, the hit ratios of the four strategies decrease when the packet loss rate increases, but CAPC always performs better than the other three strategies. From Figure 8c, hit ratio of CAPC hardly decreases as the number of receivers increases.

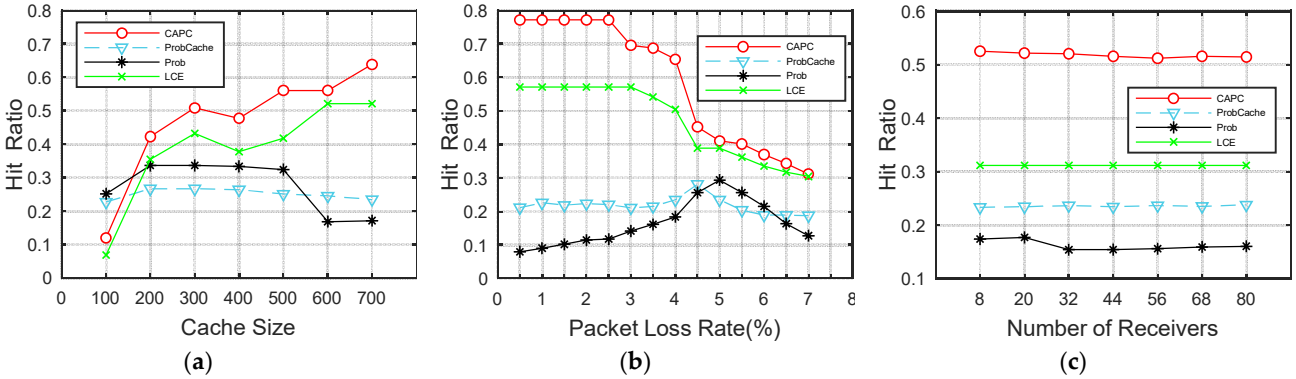


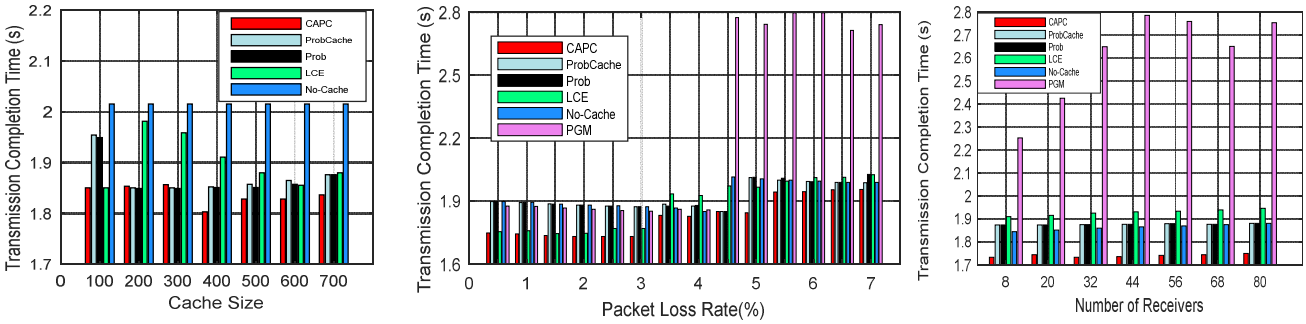
Figure 8. The comparison of average hit ratio. (**a**) The comparison of average hit ratio under different cache sizes; (**b**) the comparison of average hit ratio under different packet loss rates; (**c**) the comparison of average hit ratio under different numbers of receivers.

In summary, CAPC exhibits the highest cache hit ratio under any cache size (except 100), packet loss rate, and number of receivers. This is consistent with the conclusions in Section 6.2 that ACPC achieves the lowest loss recovery delay under various experimental conditions. This is because CAPC caches most chunks experiencing congestion loss, and most losses can be recovered from MTNs instead of the source. LCE reflects the same trend of change as CAPC, but it has a lower cache hit ratio. ProbCache and Prob have the worst performance because they cache chunks randomly regardless of whether the chunks experience loss.

6.4. Average Transmission Completion Time

In this section, we measure the average transmission completion time of all receivers for all approaches, including the time spent in the recovery phase.

Figure 9 gives the variation of average transmission completion time against different cache sizes, packet loss rates, and numbers of the receivers respectively. As shown in Figure 9a, the average transmission completion time of CAPC performs lower than the other four strategies in most cases. From Figure 9b–c, CAPC always maintains optimal performance under different packet loss rate and number of receivers, respectively. This performance comparison is consistent with the conclusions of the NLRD. In PGM, lost packets are retransmitted through multicast source, and the protocol itself generates too many NCFs. Those packets together compete with ODATAs during transmission, thereby aggravating congestion loss and increasing transmission completion time. Moreover, the situation slightly effects performance of PGM when the loss rate is small, but it is an important drawback when the loss rate is large.



(**a**) (**b**) (**c**)

Figure 9. The comparison of average transmission completion time. (**a**) The comparison of average transmission completion time under different cache sizes; (**b**) the comparison of average transmission completion time under different packet loss rates; (**c**) the comparison of average transmission completion time under different numbers of receivers.

6.5. Overhead Evaluation

Finally, we measure the overhead of all approaches during multicast data distribution, excluding the signaling overhead during multicast tree construction. Request and recovery overhead are the additional load on the MTN generated by an approach. In order to measure this parameter, the number of packets sent upstream and downstream processed by each MTN (excluding ODATAs) are counted separately, and the average values are calculated over all nodes participating in the multicast distribution tree. The results are presented as the ratio of the above average number of packets to the number of ODATAs sent by the source. Thus, the results respectively express the number of the packets sent upstream and downstream required for reliably delivering a certain number of ODATAs to a multicast group.

6.5.1. Upstream Overhead

In case of packet loss, the packets sent upstream by the MTNs and receivers are NACK packets for all approaches.

As shown in Figure 10a, the upstream overhead of CAPC, ProbCache, Prob, and LCE tends to decrease with the cache size increasing. This is because, more and more recovery packets are retransmitted by MTNs. It is worth noting that CAPC always outperforms the other four strategies except for cache size of 600. Since PGM sends packet loss feedback based on the packet-level NACK, upstream overhead of PGM is higher than the others in all experiment scenarios, and the gap gets higher as packet loss rate increases in Figure 10b. From Figure 10c, as the number of receivers increases, the upstream overhead of PGM increases. In contrast, except for PGM, the upstream overhead of other approaches is very stable as the number of multicast receivers increases. This is because they use a NACK and bitmap to collect all packet loss information of a chunk. Furthermore, the NACKs of the same chunk are aggregated at MTNs to ensure that only one NACK is sent upstream regardless of how many receivers there are. It reflects the fact that the NACK aggregation scheme has great advantages and can indeed reduce upstream overhead.

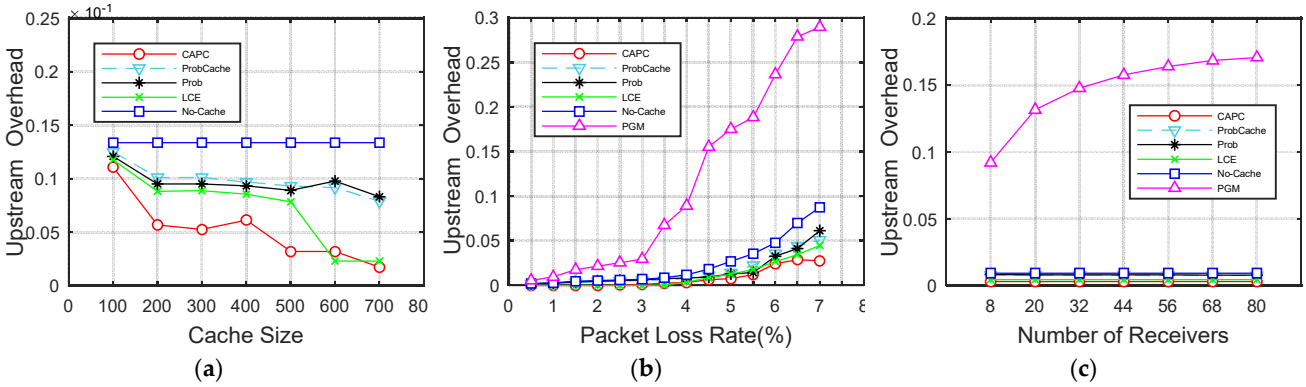


Figure 10. The comparison of upstream overhead. (**a**) The comparison of upstream overhead under different cache sizes; (**b**) the comparison of upstream overhead under different packet loss rates; (**c**) the comparison of upstream overhead under different numbers of receivers.

In particular, for CAPC, most packets can be recovered at the nearest MTN because of the higher cache hit ratio. Therefore, the upstream overhead of CAPC is the lowest in almost all cases. This is consistent with the conclusions of the previous Section 6.3.

6.5.2. Downstream Overhead

In CAPC, ProbCache, Prob, LCE, and No-Cache, packets sent downstream are the RDATAs in case of loss recovery, whereas in PGM, they are RDATAs and NCF packets. RDATAs are sent by the source in No-Cache and PGM, whereas RDATAs might be sent by the MTNs caching chunks for the other four approaches.

As shown in Figure 11a, since more chunks are cached, the downstream overhead of CAPC, ProbCache, Prob, and LCE gets lower when cache size is increasing. However, CAPC keeps the advantage over other strategies. In Figure 11b, with the increasing of packet loss rate, the amount of packet loss increases, which is bound to cause more recovery traffic, so the downstream overhead increases for all approaches. In Figure 11c, when the number of multicast receivers increases, the number of receivers connecting to the end MTN increases and the end MTN has to replicate more multicast packets, so the downstream overhead of all the approaches becomes larger. However, the downstream overhead of CAPC increases the slowest with the number of receivers increasing. Furthermore, the downstream overhead of CAPC is significantly lower than that of PGM. The overhead of PGM in the downstream direction is 3.5 to 4.6 times that of CAPC.

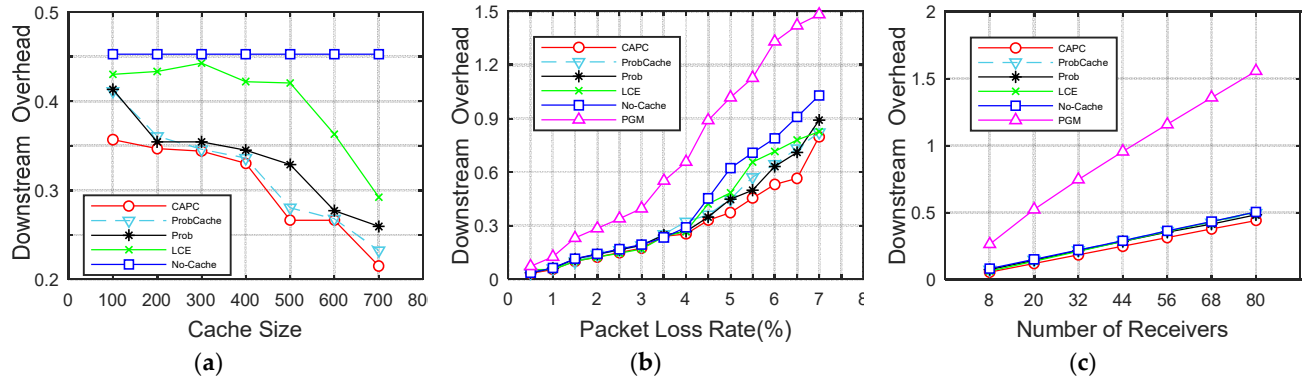


Figure 11. The comparison of downstream overhead. (**a**) The comparison of downstream overhead under different cache sizes; (**b**) the comparison of downstream overhead under different packet loss rates; (**c**) the comparison of downstream overhead under different numbers of receivers.

In general, since PGM maintains the feedback scheme in packets and uses NCF packets for feedback suppression, it generates the most download overhead traffic in all test scenarios. Other approaches use recovery isolation scheme under our proposed reliable multicast architecture. The MTNs control the propagation range of the retransmission and only forward RDATAs to the required downstream MTNs or receivers, so the downstream overhead is smaller. This result verifies the advantages of our proposed recovery isolation scheme.

7. Conclusions

In this paper, we discussed how to guarantee the reliability of multicast transmission in ICN. Firstly, we designed a reliable multicast (RM) architecture consisting of multicast tree establishment, original multicast data transmission, NACK feedback aggregation and recovery isolation scheme, and formulated problem using recovery delay as the optimization objective under the RM architecture. Secondly, to improve recovery delay performance, we proposed a Congestion-Aware Probabilistic Cache (CAPC) strategy to cache recently transmitted chunks during the original multicast data transmission. CAPC takes into account the congestion condition (indicated by congestion cost) and the cache location on the multicast tree. Then, to recover the loss more efficiently, we proposed the feedback aggregation scheme and the recovery isolation scheme to reduce overhead in feedback and recovery phase, respectively. Finally, we implemented and compared CAPC with ProbCache, Prob, LCE, No-Cache under the proposed architecture, besides comparing with classical PGM protocol. The simulation results demonstrated the significant advantages of our approach over other approaches in recovery delay, cache hit ratio, transmission completion time, and overhead.

In future work, we will combine congestion control scheme with reliable multicast to further improve the reliability and efficiency of multicast transmission. Besides, we will investigate the impact of multicast links or routers failures on multicast, and consider introducing a backup scheme for multicast links or routers to resist unreliability caused by network failures in ICN.

Author Contributions: Conceptualization, Y.D., H.N., and X.Z.; methodology, Y.D., H.N., and X.Z.; software, Y.D.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D., H.N., and X.Z.; supervision, H.N.; project administration, X.Z.; funding acquisition, H.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

Acknowledgments: We would like to express our gratitude to Jinlin Wang and Rui Han for their meaningful support for this work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chakraborty, D.; Chakraborty, G.; Shiratori, N. A dynamic multicast routing satisfying multiple QoS constraints. *Int. J. Netw. Manag.* **2003**, *13*, 321–335.
2. Lao, L.; Cui, J.H.; Gerla, M.; Maggiorini, D. A comparative study of multicast protocols: Top, bottom, or in the middle? In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; pp. 2809–2814.
3. Huang, J.; Duan, Q.; Zhao, Y.; Zheng, Z.; Wang, W. Multicast Routing for Multimedia Communications in the Internet of Things. *IEEE Internet Things J.* **2016**, *4*, 215–224.
4. Zhang, Y.; Raychadhuri, D.; Grieco, L.; Baccelli, E.; Burke, J.; Ravindran, R.; Wang, G. ICN based Architecture for IoT—Requirements and Challenges draft-zhang-iot-icn-challenges-00. ICNRG Internet-Draft. 2014. Available online: https://datatracker.ietf.org/doc/html/draft-zhang-iot-icn-challenges-00 (accessed on 22 September 2014).
5. Tsoukaneri, G.; Condoluci, M.; Mahmoodi, T.; Dohler, M.; Marina, M.K. Group communications in narrowband-IoT: Architecture, procedures, and evaluation. *IEEE Internet Things J.* **2018**, *5*, 1539–1549.

6. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376.
7. Kawasumi, R.; Hirota, Y.; Murakami, K.; Tode, H. Multicast distribution system with functions of time-shift and loss-recovery based on In-network caching and OpenFlow control. In Proceedings of the 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Compiegne, France, 28–30 October 2013; pp. 641–646.
8. Duan, J.; Xing, Y.; Zhao, G. Overview of research on caching technology in information center networks. *Comput. Eng. Appl.* **2018**, *54*, 1–10.
9. Atwood, J.W. A classification of reliable multicast protocols. *IEEE Netw.* **2004**, *18*, 24–34.
10. He, X.; Papadopoulos, C.; Radoslavov, P. Incremental deployment strategies for router-assisted reliable multicast. *IEEE/ACM Trans. Netw.* **2006**, *14*, 779–792.
11. Whetten, B.; Vicisano, L.; Kermode, R.; Handley, M.; Floyd, S.; Luby, M. Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer. IETF RFC 3048. Available online: https://tools.ietf.org/html/rfc3048 (accessed on 23 January 2020).
12. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36.
13. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A Survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1024–1049.
14. Jiang, X.; Bi, J.; Nan, G.; Li, Z. A survey on Information-centric Networking: Rationales, designs and debates. *China Commun.* **2015**, *12*, 1–12.
15. Liao, Y.; Sheng, Y.; Wang, J. Summary of Research on ICN Name Resolution Technology. *New Netw. Media Technol.* **2020**, *9*, 1–9.
16. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.C.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *Acm Sigcomm. Comput. Commun. Rev.* **2014**, *44*, 66–73.
17. Rossi, D.; Rossini, G. Caching performance of CCN under multi-path routing (and more). Technical Report, Telecom Paris-Tech: Telecom Paris-Tech, 2011.
18. NS-2 Simulator. Available online: https://www.isi.edu/nsnam/ns/ns-build.html (accessed on 4 November 2005).
19. Yoon, W.; Lee, D.; Youn, H.Y.; Lee, S. A combined group/tree approach for scalable many-to-many reliable multicast. *Comput. Commun.* **2006**, *29*, 3863–3876.
20. Adamson, B.; Bormann, C.; Handley, M.; Macker, J. NACK-Oriented Reliable Multicast (Norm) Transport Protocol. IETF RFC 5740. Available online: https://tools.ietf.org/html/rfc5740 (accessed on 12 November 2018).
21. Lin, Y.; Wu, H.; Wang, C.; Cheng, S. Dynamic Retransmission Control for Reliable Mobile Multicast. *J. Comput. Sci. Technol.* **2003**, *18*, 369–377.
22. Levine, B.; Lavo, D.; Garcia-Luna-Aceves, J. The case for reliable concurrent multicasting using shared ACK trees. In Proceedings of the Fourth ACM International Conference on Multimedia, Boston, MA, USA, 18–22 November 1996; pp. 365–376.
23. Koh, S.; Kim, E.; Park, J.; Kang, S.; Park, K.; Park, C. Configuration of ack trees for multicast transport protocols. *ETRI J.* **2001**, *23*, 111–120.
24. Paul, S.; Sabnani, K.K.; Lin, J.C.H.; Bhattacharyya, S. Reliable multicast transport protocol (RMTP). *IEEE J. Sel. Areas Commun.* **1997**, *15*, 407–420.
25. Whetten, B.; Taskale, G. An overview of reliable multicast transport protocol II. *IEEE Netw.* **2000**, *14*, 37–47.
26. Gemmell, J.; Montgomery, T.; Speakman, T.; Crowcroft, J. The PGM reliable multicast protocol. *IEEE Netw.* **2003**, *17*, 16–22.
27. Papadopoulos, C.; Parulkar, G.; Varghese, G. Error control scheme for large-scale multicast applications. In Proceedings of the IEEE INFOCOM'98, San Francisco, CA, USA, 29 March–2 April 1998; pp. 1188–1196.
28. Floyd, S.; Jacobson, V.; Liu, C.; McCanne, S.; Zhang, L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Netw.* **1997**, *5*, 784–803.
29. Wijnands, I.; Rosen, E.C.; Dolganow, A.; Przygienda, T.; Aldrin, S. Multicast using bit index explicit replication (BIER). IETF RFC 8279. Available online: https://rfc-editor.org/rfc/rfc8279.txt (accessed on 30 November 2017).
30. Desmouceaux, Y.; Clausen, T.H.; Antonio, J.; Fuertes, C.; Townsley, W.M. Reliable Multicast with B.I.E.R. *J. Commun. Netw.* **2018**, *20*, 182–197.
31. Desmouceaux, Y.; Fuertes, J.A.C.; Clausen, T.H. Reliable BIER with Peer Caching. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1826–1839.
32. Zhang, X.; Yang, M.; Wang, L.; Sun, M. An OpenFlow-Enabled Elastic Loss Recovery Solution for Reliable Multicast. *IEEE Syst. J.* **2018**, *12*, 1945–1956.
33. Mahajan, K.; Sharma, D.; Mann, V. Athena: Reliable Multicast for Group Communication in SDN-Based Data Centers. In Proceedings of the 9th International Conference on Communication Systems and Networks, Bengaluru, India, 4–8 January 2017; pp. 174–181.
34. Stais, C.; Xylomenos, G.; Voulimeneas, A. A reliable multicast transport protocol for information-centric networks. *J. Netw. Comput. Appl.* **2015**, *50*, 92–100.
35. Chen, J.; Arumaithurai, M.; Fu, X.; Ramakrishnan, K.K. Reliable Publish/Subscribe in Content-Centric Networks. In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking, Hong Kong, China, 12–16 August 2013; pp. 21–26.

36. Xin, Y.; Liu, X.; Fang, C.; Luo, H. Optimization of data retransmission algorithm in information centric networking. *Comput. Appl.* **2019**, *39*, 829–833.
37. Bian, S.; Huang, X.; Shao, Z.; Gao, X.; Yang, Y. Service Chain Composition with Failures in NFV Systems: A Game-Theoretic Perspective. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May, 2019; pp. 1–6.
38. Di Mauro, M.; Longo, M.; Postiglione, F. Availability evaluation of multi-tenant service function chaining infrastructures by multidimensional universal generating function. *IEEE Trans. Serv. Comput.* **2018**, doi:10.1109/tsc.2018.2885748.
39. Di Mauro, M.; Longo, M.; Postiglione, F.; Tambasco, M. Availability modeling and evaluation of a network service deployed via NFV. *Commun. Comput. Inf. Sci.* **2017**, *766*, 31–44.
40. Lehman, L.H.; Garland, S.J.; Tennenhouse, D.L. Active reliable multicast. In Proceedings of the IEEE INFOCOM'98, San Francisco, CA, USA, 29 March–2 April 1998; pp. 581–589.
41. Yeung, K.L.; Wong, H.L.T. Caching policy design and cache allocation for active reliable multicast. *Comput. Netw.* **2003**, *43*, 69–79.
42. Xie, F.; Feng, G.; Yang, X. Optimizing Caching Policy for Loss Recovery in Reliable Multicast. In Proceedings of the IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, Barcelona, Spain, 23–29 April 2006; pp. 1–12.
43. Li, J.; Liu, K. Network-coding-based cache policy for loss recovery enhancement in reliable multicast. *Int. J. Netw. Manag.* **2012,** *22*, 330–345.
44. Thomas, Y.; Xylomenos, G.; Polyzos, G.C. In-network packet-level caching for error recovery in ICN. In Proceedings of the 2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT), Volos, Greece, 15–19 June 2020; pp. 1–8.
45. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing Information Networking Further: From PSIRP to PURSUIT. In Proceedings of the International Conference on Broadband Communications, Networks and Systems, Athens, Greece, 25–27 October 2010; pp. 11–13.
46. Raychaudhuri, D.; Nagaraja, K.; Venkataramani, A. MobilityFirst: A robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2012**, *16*, 2–13.
47. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome Italy, 1–4 December 2009; pp. 1–12.
48. Cain, B.; Deering, S.; Kouvelas, I.; Fenner, B.; Thyagarajan, A. Internet Group Management Protocol, Version 3. IETF RFC 3376. Available online: https://datatracker.ietf.org/doc/html/rfc3376 (accessed on 10 October 2019).
49. Vida, R.; Costa, L. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. IETF RFC 3810. Available online: https://datatracker.ietf.org/doc/html/rfc3810 (accessed on 1 June 2020).
50. Handley, M.; Floyd, S.; Whetten, B.; Kermode, R.; Vicisano, L.; Luby, M. The Reliable Multicast Design Space for Bulk Data Transfer. RFC 2887. Available online: https://tools.ietf.org/html/rfc2887 (accessed on 24 August 2020).
51. Nguyen, D.; Sugiyama, K.; Tagami, A. Congestion price for cache management in information-centric networking. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 287–292.
52. Floyd, S.; Jacobson, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413.
53. Maravelakis, P.E.; Castagliola, P. An EWMA Chart for Monitoring the Process Standard Deviation When Parameters Are Estimated. *Comput. Stat. Data Anal.* **2009**, *53*, 2653–2664.
54. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic In-Network Caching for Information-Centric Networks. In Proceedings of the ACM SIGCOMM Workshop on ICN, Helsinki, Finland, 13–17 August 2012; pp. 55–60.
55. Lin, C.S.; Hwang, W. An Access Point-Based FEC Mechanism for Video Transmission over Wireless LANs. *IEEE Trans. Multimed.* **2013**, *15*, 195–206.
56. Arianfar, S.; Nikander, P.; Ott, J. On content-centric router design and implications. In Proceedings of the Co-NEXT'10: Conference on emerging Networking EXperiments and Technologies, Philadelphia, PA, USA, 30 November 2010; pp. 1–6.