Message Bundling on Structured Overlays

Kazuyuki Shudo
Tokyo Institute of Technology
Tokyo, Japan

***Abstract*—A structured overlay running as the base of a DHT or ALM occasionally receives a large number of messages collectively. Those cases are opportunities for an overlay to bundle multiple different messages into a single message and achieve extraordinarily effective message deliveries. Such message bundling reduces the number of packet transmissions on an underlay network such as an IP network. This paper presents *Collective Forwarding*, a message bundling technique for structured overlays. The technique not only reduces the number of packet transmissions but also improves throughput of message forwarding. In experiments, the number of packet transmissions and the time to get items were reduced to 12% and 9.7% respectively at best. Theoretical analysis matches and then supports the experimental results.**

***Keywords*-peer-to-peer, structured overlays, DHT, message bundling**

I. Introduction

A structured overlay is an application-level network, by which autonomous nodes perform ID-based message delivery without central servers. On top of it, higher-level services such as distributed hash tables (DHTs), application-layer multicast (ALM) are built and work [1]. Those services are steady bases of highly scalable and resilient distributed systems. For example, their application to DNS [2] has been throughly studied and content delivery systems with DHTs [3], [4] have been deployed.

A message delivery on a structured overlay with N nodes generally requires $O(logN)$ times message forwarding. Each forwarding is a packet delivery on an underlay network such as an IP network. A message delivery on an overlay is heavier and takes more time than a packet delivery on an underlay because it consists of multiple packet deliveries.

A structured overlay supporting a DHT or ALM occasionally receives a large number of messages collectively. A DHT generates a large number of messages at a time when it is filled up for its initial use, backed up and restored. For example, DNS [2], RFID databases [5], [6], [7] and other IoT cases store millions or more data items. Those cases are opportunities for an overlay to achieve extraordinarily effective message deliveries by bundling multiple different messages into a single message.

This paper presents *Collective Forwarding*, a message bundling technique for structured overlays. It alleviate an underlay network such as an IP network and improve efficiency of message delivery on structured overlays. Assume that a node delivers multiple different messages to their responsible nodes. If delivery routes of those messages overlap each other, the number of message forwarding is reduced by bundling such messages into a single message. Such bundling improves network throughput and mitigates the load of nodes on an overlay that forwards messages. It also reduces the number of packet transmissions and alleviates routers on an underlay.

It is possible to promote such route overlaps not only just expecting unintentional overlaps to happen. We can increase overlapping opportunities by grouping messages into sets of messages that possibly share their routes (Section IV-A).

This paper is an extended version of our previous work [8]. The extension includes theoretical analysis (Section III), new experimental results and detailed analysis of the results (Section IV-B).

The rest of this paper is organized as follows. Section II describes the proposed technique, Collective Forwarding. Section III shows theoretical analysis of effect of the technique. In Section IV, we evaluate the technique by experiments. Section V shows related work and we conclude in Section VI.

II. Collective Forwarding

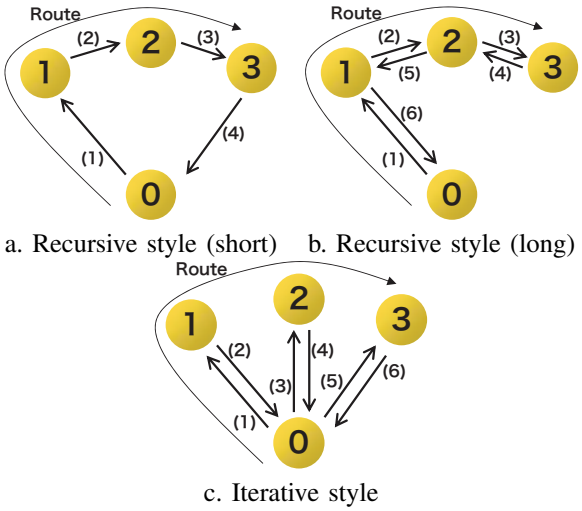
In this section, we describe *Collective Forwarding*, a message bundling technique for structured overlays.

A structured overlay delivers a message to responsible nodes that are responsible for the target ID of the message. Each node on the delivery route determines a next hop by referring to its routing table. And the node forwards the message to the next hop by requesting its underlay network such as an IP network to do it. With the iterative style of forwarding (Figure 1c) [9], a node on a route does not directly forward a message to the next hop and instead the originating node 0 communicates with all nodes. In this paper, we also call such redirections from node 1 to node 2, and from node 2 to node 3 in Figure 1c forwarding.

Assume that a node has multiple messages to be delivered. Each message has its own target ID. A node can forward the messages at a time if the next hops of those messages are the same node. With the proposed technique, a node groups such messages into a single combined message and forwards it to the common next hop. Such bundling reduces the number of message forwarding. The bundling improves network throughput by combining multiple fine-grained messages into a coarse bundle. It mitigates the load of nodes on an overlay that forwards messages by reducing the number of forwarding. As for an underlay network, the technique reduces the number of packet transmissions and alleviates routers. We call the proposed technique Collective Forwarding.

Figure 2 summarizes forwarding processes with the proposed technique. The left figure is a case without the proposed

Proc. 22nd IEEE Symposium on Computers and Communications
(IEEE ISCC 2017), pp.424-431, July 2017



a. Recursive style (short) b. Recursive style (long)

c. Iterative style

Fig. 1. Forwarding styles.

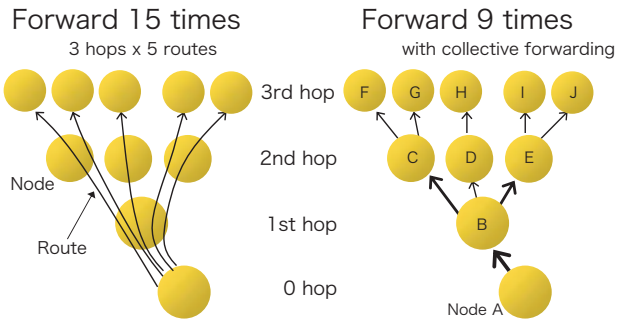


Fig. 2. Reduction in the number of message forwarding.

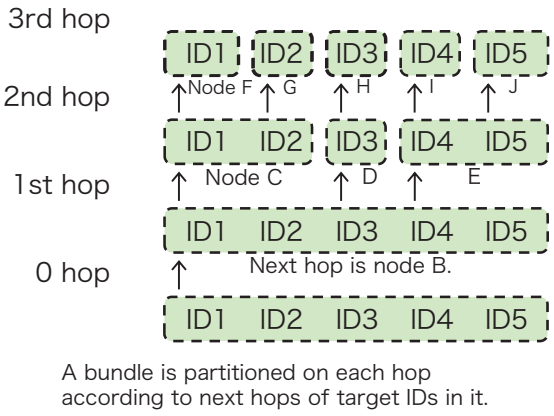


Fig. 3. Bundle partitioning according to next hops.

technique and messages are delivered one by one, and the right figure shows a case with the technique. The originating node sends 5 messages and the length of the all routes is 3. The usual and individual deliveries, shown in the left figure, requires 15 times of forwarding. The number of forwarding is reduced to 9 by the proposed technique.

The procedure of the proposed technique is as follows. We call the combined message or target IDs of each messages in it "bundle".

1) The originating node puts all target IDs into a bundle.
2) A node on the route determines next hops for all target IDs in the bundle.
3) The node divides the bundle into sub-bundles. It means that the node puts the target IDs whose next hops are the same into the same resulting sub-bundle.
4) The node forwards the resulting sub-bundles one by one.
5) Returns to 2.

Figure 3 shows how a bundle is divided on each hop of routing. Node A, the originating node puts all target IDs into a single bundle. Node A then determines next hops for all the target IDs, sees that it is node B for all the five target IDs, and forwards the bundle to node B without dividing it. Node B determines that the next hops are node C, D and E, divides the given bundle into 3 sub-bundles, and forwards the three sub-bundles to each next hop.

The proposed technique improves network throughput and lightens the load of nodes on an overlay that forwards messages. It is generally difficult to achieve high throughput with fine-grained communications. For example, as long as using TCP or UDP, throughput benchmarks do not show the peak performance of the IP network with a small communication unit such as 40 byte. The reasons include high interruption rate on a computer and header overhead of TCP and UDP. The proposed technique improves throughput by combining messages.

In forwarding, a node on an overlay interprets a received message and construct a message to be sent. The number of times those processes occur is proportional to the number of forwarding. The load of the processes is noticeable though it depends on the protocol of an overlay and any discussion on it requires more concrete numbers about it. The proposed technique lightens the load of nodes by reducing the number of forwarding, or at least keeps the same load as it without the technique.

As for underlay network, the technique lightens the load of routers by reducing the number packet transmissions. For example, in a forwarding process on an IP network, a router determines the output port by analyzing the header of an IP packet. The bottleneck of the forwarding process is such analyzing process required for each IP packet. Because of it, the forwarding performance of a router is represented by packets per second (pps). The proposed technique lightens the load of routers by reducing the number of packets.

Note that traffic remains the same though the bundling technique reduces the number of packet transmissions. The route and the traffic of every message on an underlay network do not change because the messages are just combined. The total size of headers except target IDs decreases by the bundling, but its impact is limited because target IDs and bodies are dominant in size. But the proposed technique effectively reduces traffic of application-layer multicast (ALM). In ALM, the identical message body is delivered to all receiving nodes. A bundle has just a single message body for all the receiving nodes.

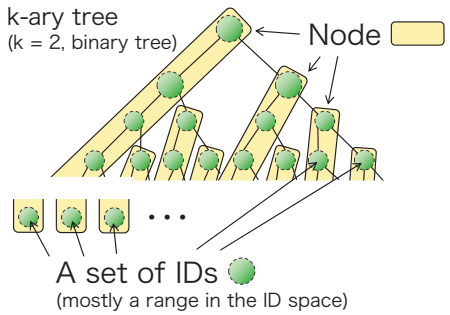


Fig. 4. An analytic model of routes originated in a node.

TABLE I
RATIOS OF THE NUMBER OF PACKET TRANSMISSIONS WITH COLLECTIVE FORWARDING TO ONE WITHOUT THE TECHNIQUE.

	"random"	"clustered"
Theoretical ratios (in Figure 5):		
Binary tree	0.766	0.102
Experimental ratios (in Figure 7):		
Chord	0.78 $\sim$ 0.80	0.18
Theoretical ratios:		
16-ary tree	0.893	0.102
Experimental ratios:		
Pastry	0.94 $\sim$ 0.95	0.32 $\sim$ 0.34
Tapestry	0.92	0.12

A. Initial bundle grouping

A large bundle is prone to cause reliability problems on an unreliable underlay. On an IP network, a packet larger than maximum transmission unit (MTU) is fragmented to multiple pieces. In that case, loss of one of the pieces causes loss of the original large packet holding a large bundle.

The best size of a bundle depends on an underlay. We choose 10 as the number of messages in a bundle for the following analysis and experiments.

III. THEORETICAL ANALYSIS

How much does the technique reduce the number of packet transmissions on an underlay? Figure 4 shows an analytic model. It is a k-ary tree. With a k-ary tree, it is possible to model message forwarding on structured overlays. For example, a binary tree ($k = 2$) suits Chord [10] and other algorithms based on it such as Chord$^{\#}$ [11] and FRT-Chord [12]. A 2^b-ary tree suits algorithms based on Plaxton prefix routing [13] with the base $k = 2^b$ including Pastry [14] and Tapestry [15].

The root of a tree corresponds a node sending messages and leaves of a tree correspond nodes that the messages are delivered to. A message is forwarded along a path from the root to a leaf.

To be precise, a vertex of the k-ary tree is a set of IDs. In Figure 4, ID sets delivered from a node to the node itself are drawn in the same node. It is the reason why a single node (yellow box) has multiple ID sets (green balls). The ID set of a vertex with children is the union of ID sets of all the children. ID sets of all the children are disjoint. The union of ID sets of all the leaves covers the entire ID space.

A node on an overlay corresponding to a leaf is responsible for the set of IDs of the leaf. A node corresponding to a vertex also corresponds one of the children of the vertex. In Figure 4, a leftmost child is such a child. A step downward to such a child occurs in the node and does not cause message forwarding.

Now we start counting the number of packet transmissions on an underlay. It equals to the number of bundle forwarding without packet fragmentation (Section II-A). We avoid packet fragmentation by limiting the size of a bundle. For simplicity, we assume a perfect k-ary tree, in which all leaf nodes are on the same level. Probabilities of message deliveries to all the leaves are equal.

Figure 5 shows ratios of the number of packet transmissions with Collective Forwarding to one without the technique. In all the figures in this paper, "serial" means message forwarding without the technique, "random" and "clustered" means bundle forwarding with the technique. "Random" indicates initial bundle grouping (Section II-A) at random and "clustered" indicates grouping target IDs with the procedure described in Section IV-A. The size of initial bundles is 10 as well as all the experiments in Section IV.

We focus on 1000 nodes, that is the same as all the experiments in Section IV. Table I shows a summary of the following values. Theoretical ratios of a binary tree shown in Figure 5 a are 0.766 in "random" and 0.102 in "clustered". Experimental ratios of Chord, to which a binary tree conforms, shown in Figure 7 are from 0.78 to 0.80 in "random" and 0.18 in "clustered". The values of a 16-ary tree, Pastry and Tapestry are also derived as above.

The theoretical values do not involve messages for overlay maintenance but the experiments involve them. Such messages weaken the reduction effect. It is the major reason for the worse values of the experimental ratios than theoretical ratios. But the number of packet transmissions in "random" is larger than "clustered" and it hides messages for overlay maintenance. Because of it, in "random", theoretical ratios provided better approximations of experimental ratios than "clustered".

IV. EVALUATION

We evaluate the proposed technique, Collective Forwarding by measuring number of packet transmissions and time to deliver messages.

We implemented he technique in Overlay Weaver [16], [17] and carried out the experiments with it. Overlay Weaver is an implementation of structured overlays, which performs ID-based message delivery and provides DHT and ALM functions on it. The proposed technique is always enabled and works for every message delivery in Overlay Weaver because it is a natural extension of structured overlays (Section V).

We carried out all experiments with the distributed environment emulator Overlay Weaver provides by running and controlling multiple DHT shell instances on it. The emulator hosts massive nodes on a single computer and controls those

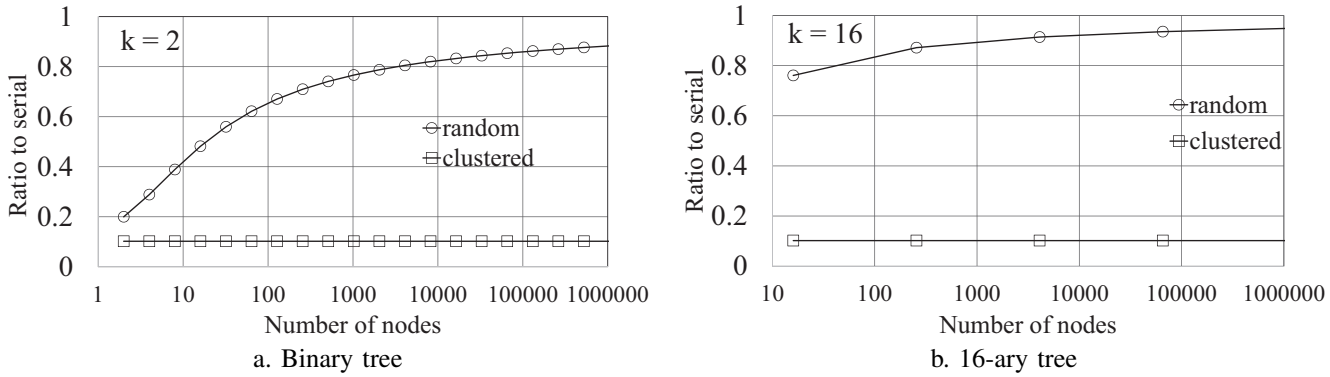


Fig. 5. Ratio of the number of message transmissions with Collective Forwarding to the case without the technique.

nodes along the given emulation scenario. The code running on the emulator also works on a real network. Because of it, the implementation evaluated in this section run on PCs and Internet as it is.

We used a PC with a 2.8 GHz Intel x86 processor, Linux 2.6.25 for x86-64 and HotSpot Server VM of Java 2 SE 5.0 Update 15. The version of Overlay Weaver is 0.8.7. All experiments were made 5 times and we adopted the average of middle 3 values as the result.

A. Initial bundle grouping

The proposed technique takes much effect by grouping the initial bundles as routes for target IDs in a bundle overlap each other for the most part (Section II-A).

We gave priority to have good results of the initial grouping over efficiency of the grouping process itself because the purpose of the experiences is to investigate Collective Forwarding. Along the policy, we adopted a simple grouping procedure as follows though it takes much time.

1) Provides an empty bundle. Sets the ID last added to the previous bundle as the mark ID. For the first bundle, sets 0x00...0 as the mark ID.
2) Takes the ID closest to the mark ID from not-processed IDs, and moves it to the bundle.
3) Takes the ID whose average distance to all IDs in the bundle is the smallest in not-processed IDs, and moves it to the bundle
4) Repeats 3. unless the size of the bundle (the number of IDs) reaches the given limit. Fixes the bundle if the size reaches the limit.
5) Returns to 1.

This procedure is expected to produce a good grouping result by choosing the ID which looks the best ID to be added to the bundle.

Distance here is the numerical distance between IDs, which is calculated in a way dependent on each routing algorithm. For example, the distance from 3 to 4 is 1 in Chord, but $3 \oplus 4 = 7$ in XOR distance of Kademlia. It follows that effective grouping of initial bundles depends on each routing algorithm even though the proposed technique and its implementation in Overlay Weaver are neutral to routing algorithms.

Note that known clustering algorithms cannot be applied to the initial bundle grouping because ID distance on a structured overlay depends on a direction as shown in the previous paragraph. Chord is a typical example and the distance from 3 to 4 is 1 but it is $2^{160} - 1$ (in case of 160 bit ID) from 4 to 3.

B. Number of packet transmissions

This section shows how many packet transmissions were reduced by the proposed technique. We provided an emulation scenario which constructs an overlay with 1000 nodes, puts 50000 data items on the overlay and gets them, and run the scenario with the distributed environment emulator. The size of a bundle is 10 and initial bundles were grouped by the procedure shown in Section IV-A before the experiment.

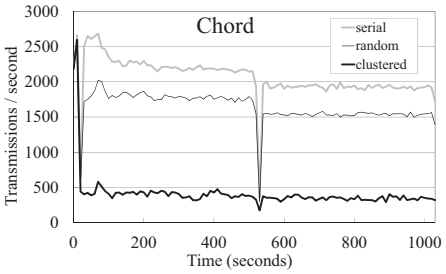
The emulation scenario is as follows. It takes 1040 seconds after nodes start joining the overlay.

1) Starts 1000 nodes.
2) Constructs an overlay by letting the nodes join the overlay every 20 millisecond.
3) Sleeps for 10 seconds.
4) Puts 50000 data items, one every 10 millisecond.
5) Sleeps for 10 seconds.
6) Gets 50000 data items, one every 10 millisecond.

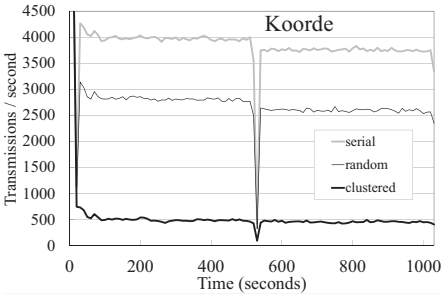
Nodes to which put and get were determined at random when the scenario is generated. The scenario puts and gets a bundle every 100 millisecond and it corresponds to putting/getting a data item every 10 millisecond.

Figure 6 shows the number of packet transmissions a second between nodes on an underlay network. Figure 6 a, b, c, d and e are the results with routing algorithms, Chord [10], Koorde [18], Pastry [14], Tapestry [15] and Kademlia [19] respectively. The size of a digit b in Pastry and Tapestry is 4 bit and then $k = 2^4 = 16$. The forwarding style (Figure 1) here is iterative but experiments with recursive style showed similar results.

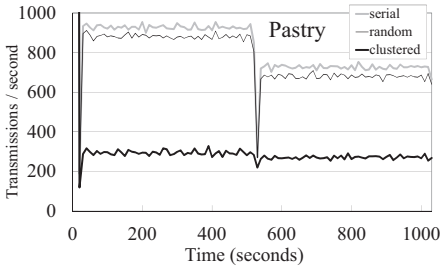
Figure 7 is about the total number of message transmissions required to put and get data items. The numbers do not include messages to construct the overlay because it is not affected by the proposed technique. The numbers in Figure 7 is ratios of the number of transmissions with the technique to one without the technique.



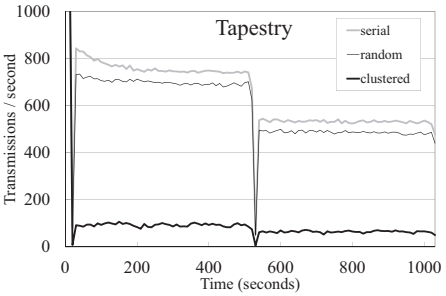
a. Chord



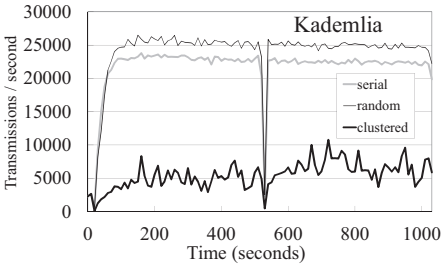
b. Koorde



c. Pastry



d. Tapestry



e. Kademlia

Fig. 6. Number of packet transmissions on an underlay to deliver messages on an overlay.

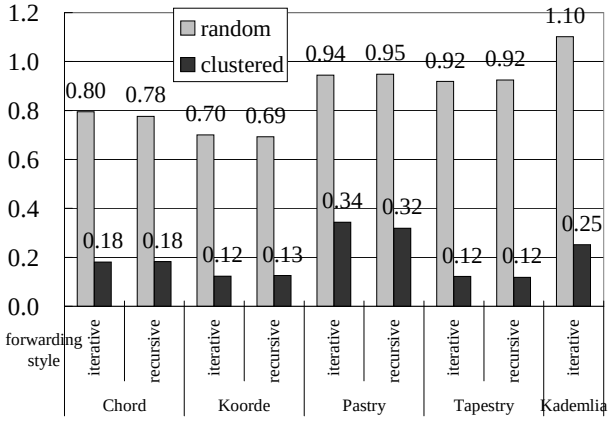


Fig. 7. Experimental ratio of the number of packet transmissions with Collective Forwarding to the case without the technique.

In the figures, meanings of "serial", "random" and "clustered" are the same as Section III.

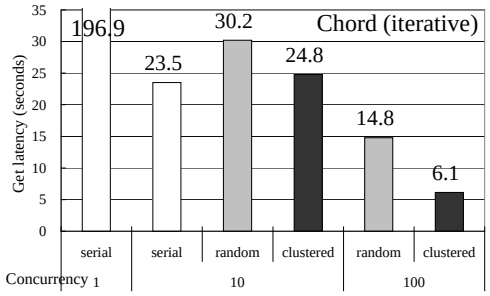
The proposed technique with the initial bundle grouping ("clustered") reduced the number of transmissions to 34% in Pastry $\sim$ 12% in Koorde and Tapestry (Figure 7).

In Pastry, a large number of messages to maintain the overlay weaken the reduction effect. Pastry, at least its implementation for Overlay Weaver involves more messages to maintain an overlay than other algorithms. Because of the maintenance messages, the number of transmissions just after 500 second in Figure 6c decreased but more than 200 transmissions / second remained.

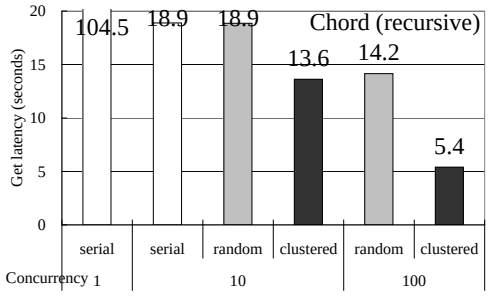
The number of transmissions for get operations is smaller than put operations, except "clustered" in Kademlia. It is due to Overlay Weaver protocol. An originating node completes a get operation by receiving a reply from a responsible node. But in a put operation, an originating node sends another put request to a responsible node after receiving the reply from it. The protocol prevents a possibly large value being forwarded along the route. This difference is conspicuous with Pastry shown in Figure 6c because it becomes larger with shorter routes and Pastry tends to result in shorter routes than other algorithms Overlay Weaver provides.

The number of transmission decreased with the technique even though the initial bundle grouping is at random ("random"), except Kademlia. It includes the reduction by occasional overlaps of routes in a bundle.

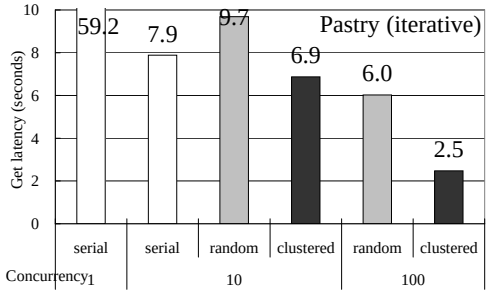
In Kademlia, the number of transmissions increased in "random" from "serial" (Figure 7), and get operations required more number of transmissions than put operations (Figure 6e). It is due to the routing table maintenance protocol of Kademlia. With the protocol, a node knows another node by communicating with the another node and try to store the another node in its routing table. Once part of its routing table, a k-bucket, is filled, it sends and receives more messages to check aliveness of other nodes in its routing table. Because of it, much communication on a node yields more communication. The proposed technique concentrates communication on limited part of nodes. A node initiating a message was



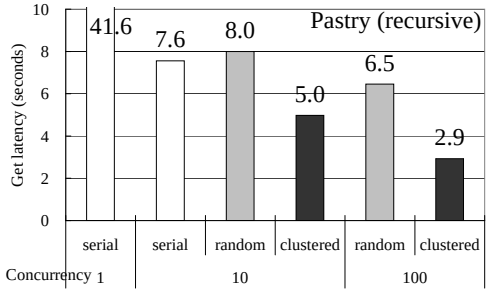
a. Chord (iterative)



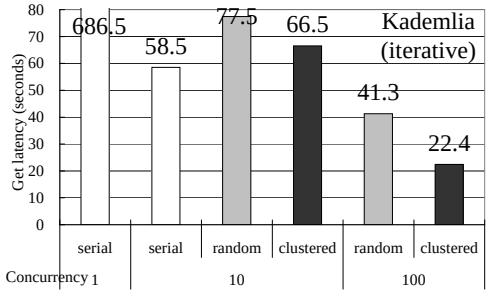
b. Chord (recursive)



c. Pastry (iterative)



d. Pastry (recursive)



e. Kademlia (iterative)

Fig. 8. Time to deliver 10000 messages.

chosen randomly 50000 times without the technique ("serial"), but such a node was chosen randomly 5000 times with the technique and a chosen node performed more communication.

C. Time to deliver messages

This section shows how long time it took to get data items from a DHT. The distributed environment emulator ran an emulation scenario which constructs a DHT with 1000 nodes and get 10000 data items from the DHT. Emulated communication latency on the underlay network is 1 millisecond. These parameters emulate a large-scale distributed database on a local-area network (LAN) for applications such as RFID databases [5], [6], [7]. Communication latency of today's LAN media such as 10Gb Ethernet and InfiniBand is around microseconds but software stacks such as TCP/IP stacks and DHT implementations cause larger latency by one or more orders of magnitude. Because such software-originated latency varies and heavily depends on software, we chose uniform communication latency to investigate solely an impact of the proposed technique. The unit, millisecond, has no meaning.

Figure 8 shows the time required to get 10000 data items. As same as Figure 6 and Figure 7 (Section IV-B), meanings of "serial", "random" and "clustered" are the same as Section III. Concurrency in the graphs is the number of message delivery (i.e. get) requests which can be processed concurrently on an overlay. Concurrency 10 with "random" and "clustered" means delivery of bundles one by one while a bundle includes 10 target IDs. In concurrency 10 with "serial", a scenario sequencer of the emulator connected to 10 nodes in the DHT and requested up to 10 delivery requests concurrently. Concurrency 100 means requesting up to 10 bundles to be delivered through 10 connections while a bundle includes 10 target IDs. It is not obvious which of concurrency 10 and concurrency 100 with the proposed technique ("random" and "clustered") is appropriate to be compared with concurrency 10 without the technique ("serial"). The former is with the same concurrency but the latter utilizes the same number of connections, 10.

The technique with the initial bundle grouping ("clustered" with concurrency 10) reduced the time to 13.0% in Chord with recursive style $\sim$ 9.7% in Kademlia from "serial" with concurrency 1. These reduction rates are comparable with those without the technique with concurrent requesting ("serial" with concurrency 10), which are 18.2% in Pastry with recursive style $\sim$ 8.5% in Kademlia. The proposed technique with concurrent requesting (concurrency 100) could reduced the time further to 7.03% in Pastry with recursive style $\sim$ 3.12% in Chord with iterative style.

V. Related work

MARIF [20], [21], a message bundling technique for DHTs, was proposed just after our previous work [8]. Collective Forwarding is an extension to structured overlays in contrast with MARIF, that is a data distribution protocol dedicated to DHTs. A message delivery and forwarding process of a single message in Collective Forwarding is just the same as a plain structured overlay while MARIF requires two phases as key

range examination and data transfer. Because of it, Collective Forwarding works with ALM protocols such as SCRIBE [22] and improves their performance though MARIF does not.

In MARIF, when a node tries to send a large number of messages, it examines the key range that another node is responsible for by asking the another node. Then it sends multiple messages that the another node is responsible for to the another node. In MARIF, a node can send unlimited number of messages continuously as a data stream to their responsible node while Collective Forwarding cannot. It contributes highly efficient data distribution by MARIF.

Collective Forwarding can be recognized as a multicast protocol (Section II) such as IP multicast and application-layer multicast (ALM). Every protocol for IP multicast constructs a delivery tree that consists of relaying and receiving routers before actual packet delivery. And a packet is delivered to all the routers along the tree. Unstructured overlay-based ALM protocols deliver a message to the whole overlay. They are classified to push-based and pull-based. Push-based protocols [23] construct a delivery tree. Pull-based protocols [24], [25] can broadcast to any form of an overlay even if it is not a tree. Structured overlay-based ALM protocols [22], [26] construct a delivery tree for a group of receiving nodes based on a structured overlay. Our protocol is different from the existing multicast protocols as it delivers a message to nodes specified by the sending node. Because of it, our protocol does not require a delivery tree.

Bundling multiple packets into a single packet is a common technique to improve communication performance. Bundling techniques have been studied, for example, for wireless sensor networks [27], delay tolerant networks [28], and virtual machines [29]. They do in-network processing in a lower layers and their concern is efficiency in a specific layer. Our target is an overlay network, that works over its underlay network and has an independent topology from the underlay. In case of an overlay, efficiency heavily involves its underlay and our proposal and experiments focus on it.

There are techniques to improve efficiency of delivery of a single message while our target is delivery of multiple messages. Those techniques reduce latency of forwarding or/and shorten the length of a forwarding path, a route. Proximity routing [30], [31] is an example of the former attempts. The latter includes attempts to keep the number of forwarding $O(1)$ times [32] and keep it once [33], [34].

VI. Summary

In this paper, we proposed Collective Forwarding, a message bundling technique for structured overlays. A node forwards multiple messages collectively as a bundle if those messages' next hops are identical. It alleviates an underlay network and improve throughput of message forwarding. The technique reduces the number of forwarding on an overlay and consequently the number of transmission on an underlay.

We analyzed the technique theoretically, implemented it in Overlay Weaver and conducted experiments with various routing algorithms. The number of packet transmissions and the time to deliver messages were reduced to 12% and 9.7% respectively at best. With concurrent requesting, the time to deliver messages could be further reduced to 3.12% at best.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 25700008 and 16K12406. This work was also supported by New Energy and Industrial Technology Development Organization (NEDO) especially in detailed comparison with existing techniques.

References

[1] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a common API for structured peer-to-peer overlays," in *Proc. IPTPS'03*, Feb. 2003.

[2] V. Pappas, D. Massey, A. Terzis, and L. Zhang, "A comparative study of the DNS design with DHT-based alternatives," in *Proc. INFOCOM 2006*, Apr. 2006.

[3] "BitTorrent," http://www.bittorrent.com/.

[4] "Vuse (formerly known as Azureus)," http://www.vuse.com/.

[5] Y. Doi, S. Wakayama, M. Ishiyama, S. Ozaki, and A. Inoue, "On scalability of DHT-DNS hybrid naming system," in *Proc. AINTEC 2006*, Nov. 2006, pp. 16–30.

[6] Y. Doi, S. Wakayama, and S. Ozaki, "A design for distributed backup and migration of distributed hash tables," in *Proc. SAINT 2008*, Jul. 2008, pp. 213–216.

[7] L. Schmidt, N. Mitton, D. Simplot-Ryl, R. Dagher, and R. Quilez, "DHT-based distributed ALE engine in RFID middleware," in *Proc. IEEE RFID-TA 2011*, Sep. 2011, pp. 319–326.

[8] K. Shudo, "Collective forwarding on structured overlays," *IPSJ Transactions on Advanced Computing Systems*, vol. 2, no. 3, pp. 39–46, Sep. 2009 (in Japanese).

[9] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. USENIX '04*, Jun. 2004.

[10] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[11] T. Schütt, F. Schintke, and A. Reinefeld, "Range queries on structured overlay networks," *Computer Communications*, vol. 31, no. 2, pp. 280–291, Feb. 2008.

[12] H. Nagao and K. Shudo, "Flexible Routing Tables: Designing routing algorithms for overlays based on a total order on a routing table set," in *Proc. IEEE P2P'11*, Aug. 2011, pp. 72–81.

[13] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," *Theory of Computing Systems*, vol. 32, no. 3, pp. 241–280, Jun. 1999.

[14] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Middleware 2001*, Nov. 2001.

[15] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *Journal on selected area in communications*, vol. 22, no. 1, pp. 41–53, Jan. 2004.

[16] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay Weaver: An overlay construction toolkit," *Computer Communications (Special Issue on Foundations of Peer-to-Peer Computing)*, vol. 31, no. 2, pp. 402–412, Feb. 2008.

[17] "Overlay Weaver: An overlay construction toolkit," http://overlayweaver.sf.net/.

[18] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proc. IPTPS'03*, Feb. 2003.

[19] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. IPTPS'02*, Mar. 2002.

[20] K. Mizutani, T. Mano, O. Akashi, and K. Fukuda, "Efficient query bundling mechanism in a DHT network," in *Proc. IEEE GLOBECOM 2012*, Dec. 2012, pp. 2695–2700.

[21] ——, "MARIF: Multiple queries look-up architecture using range information feedback in a DHT network," *IEICE Transactions on Communications*, vol. E96-B, no. 7, pp. 1680–1690, Jul. 2013.

[22] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowston, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 8, pp. 1489–1499, 2002.
[23] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 8, pp. 1456–1471, 2002.
[24] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proc. INFOCOM 2005*, Mar. 2005.
[25] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. IPTPS 2005*, Feb. 2005.
[26] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment," in *Proc. SOSP'03*, Oct. 2003.
[27] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Proc. SOSP 2001*, Oct. 2001, pp. 146–159.
[28] W.-B. Pöttner and L. Wolf, "Opportunistic data aggregation in delay tolerant networks," in *Proc. IEEE ISCC 2015*, Jul. 2015, pp. 840–845.
[29] M. Bourguiba, K. Haddadou, I. E. Korbi, and G. Pujolle, "Improving network I/O virtualization for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 673–681, Mar. 2014.
[30] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. SIGCOMM 2003*, Aug. 2003.
[31] T. Miyao, H. Nagao, and K. Shudo, "A method for designing proximity-aware routing algorithms for structured overlays," in *Proc. IEEE ISCC'13*, 2013.
[32] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Rennesse, "Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead," in *Proc. IPTPS'03*, Feb. 2003.
[33] A. Gupta, B. Liskov, and R. Rodrigues, "Efficient routing for peer-to-peer overlays," in *Proc. NSDI '04*, Mar. 2004, pp. 113–126.
[34] Y. Ando, H. Nagao, T. Miyao, and K. Shudo, "FRT-2-Chord: A DHT supporting seamless transition between one-hop and multi-hop lookups with symmetric routing table," in *Proc. ICOIN 2014*, 2014.

Proc. 22nd IEEE Symposium on Computers and Communications
(IEEE ISCC 2017), pp.424-431, July 2017