

Article

Topology-Optimal Deployment of Operating Systems for a Cluster Supercomputer

Wan Yeon Lee ¹, Jinseung Ryu ^{2,*}, Seungwoo Rho ², Sangwan Kim ² and Kimoon Jeong ² ¹ Department of Computer Science, Dongduk Women's University, Seoul 02748, Republic of Korea² Center for Development of Supercomputing System, Korea Institute of Science and Technology Information (KISTI), Daejeon 34141, Republic of Korea; kmjeong@kisti.re.kr (K.J.)

* Correspondence: jsu@kisti.re.kr; Tel.: +82-42-869-0659

Abstract

Diskless computing nodes in cluster supercomputers require massive deployment procedure of operating systems software for their booting. Most previous studies for massive deployment of large files focus on developing flexible and scalable mechanisms in hidden or changeable network topologies, but the topology-optimal deployment has been rarely investigated in fixed open network topologies. In this article, we investigate the topology-optimal deployment method designed for a specific cluster supercomputer with the fat-tree topology. We examine possible deployment approaches and propose the optimal peer-to-peer deployment method. The proposed deployment method minimizes the number of peer-to-peer transmission steps by fully utilizing resources of the fat-tree topology and avoids contention between peer-to-peer communications. Simulation results show that the proposed method completes deployment at least nine times faster than the baseline method.

Keywords: cluster supercomputer; massive deployment; peer-to-peer; fat-tree; network topology

1. Introduction

In cluster supercomputers consisting of many computing nodes, a fast deployment procedure of an operating system software or big-size application software is important for efficient utilization of cluster supercomputers. Many studies have been progressed for rapid, flexible and scalable deployment of operating system software or big-size application software. However, the optimal deployment method has been rarely handled.

There are three approaches for wide deployment of massive data: unicast-based, broadcast-based, and multicast-based. The unicast-based approach [1,2] operates based on the data transmission between one sender node and one receiver node, and it is designed according to the TCP protocol in which there is no data transfer loss with a handshaking transfer confirmation. The broadcast-based approach [3] operates based on the data transmission between one sender node and the other all nodes, and it is designed according to the UDP protocol in which data transfer loss may occur due to absence of transfer confirmation procedure. The multicast-based approach [4,5] operates based on the data transmission between one or many sender nodes and many receiver nodes, and it is designed according to the UDP protocol similar to the broadcast-based approach. The unicast-based method has been studied more actively than the broadcast-based and the multicast-based methods, because the reliable data transfer without loss during transmission is very critical for essential software of computer operation.



Academic Editor: Yutaka Ishibashi

Received: 5 January 2026

Revised: 31 January 2026

Accepted: 2 February 2026

Published: 4 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

In this study, we focus on the unicast-based approach and exclude the broadcast-based and the multicast-based approaches due to their low reliability issue of data transfer. In the unicast-based approach, if only one node plays the sender and the other nodes always play the role of the receiver, it consumes a lot of times. Therefore, in most of previous unicast-based studies, the receiver node plays the role of sender in the next step, referred to as the peer-to-peer (P2P) mechanism [6]. Most of them deal with flexible and scalable deployment and do not consider topology-specific deployment. Although a few previous studies [7–9] addressed that topology-aware deployment affects the performance, they did not handle the topology-optimal deployment method.

There is a tradeoff between fast deployment design and scalable (or flexible) design. If the deployment consumes too much time in supercomputer, rapid initial deployment of the operating systems file is a sensitive issue to users. The deployment time of the baseline method currently used in the Nurion supercomputer consumes about two hours. So, the new design of the fastest deployment in the supercomputer with long deployment time is more critical than the design of scalable or flexible deployment.

In this article, we propose the optimal deployment method for a specific cluster supercomputer, called *Nurion*. The Nurion supercomputer consists of about eight thousand computing nodes connected with the fat-tree topology of Intel switches. The proposed method fully utilizes resources of the fat-tree topology and minimizes contention between peer-to-peer communications. The contributions of this study are as follows.

It is shown that the proposed method finds the maximal paths concurrently transmittable from senders to receivers without contention and distributes evenly the found paths to senders in a target fat-tree structure.

It is verified through mathematical analysis that the proposed method is optimal in terms that it requires the minimum steps of peer-to-peer unicast for completion of deployment broadcasting under an idealized transmission model with uniform link cost regardless of number of traversed links.

Simulation experiment results show that the proposed scheme completes the deployment faster than the previous baseline method by at least nine times.

The rest of this paper is organized as follows. In Section 2, related previous studies are reviewed. Section 3 describes the considered system model and the detailed working process of the proposed scheme. Section 4 shows the evaluation results of the proposed scheme with mathematical analysis and simulation experiments. Lastly, Section 5 summarizes the study and discusses future research directions.

2. Related Works

The deployment procedure of large files in cluster or distributed computers is designed with goals of rapid initial deployment, operational flexibility, or massive scalability. The goals are selected according to the system requirements. In distributed or cloud computer systems, massive scalability and operation flexibility are preferred rather than rapid initial deployment. On the contrary, in cluster computer systems, rapid initial deployment is preferred rather than massive scalability or operational flexibility.

For reliable file deployment essential to safe computer operation such as software file of operating systems, the unicast-based communication mechanism is preferred rather than the broadcast-based or multicast-based communication mechanisms. The unicast-based communication mechanism utilizes the TCP protocol [10] in which there is no data transfer loss with a handshaking transfer confirmation, while the broadcast-based and multicast-based communication mechanisms utilize the UDP protocol [10] in which data transfer loss may occur due to absence of transfer confirmation procedure. Most of deployment methods based on the unicast-based communication employ peer-to-peer technology [6]

in which the receiver node plays the role of sender in the next step. BitTorrent [11] and eDonkey [12] are widely used popular peer-to-peer protocols.

In the cloud or Internet distributed system, the network topology is usually hidden or changeable. Hence, previous studies [13–16] for the deployment mechanism of large files in the cloud or Internet distributed system consider the network topology as a hidden and modifiable network. The previous study [13] introduces the adoption of peer-to-peer technology to distributed infrastructure for the management and storage access of large files. The previous study [14] presents a light deployment method without creating any image file or using external storage in the deployment process, whereas existing methods create image files or employ external storage. The previous study [15] presents an experimental and analytical performance study over peer-to-peer networks for large-scale content distribution over the Internet. This study examines the strength of peer-to-peer technology in high-speed networks, identifies bottleneck points of performance, and quantifies the requirements in various scenarios. The previous study [16] employs peer-to-peer technology for cloud infrastructures in order to speed up the service provisioning. A BitTorrent-like peer-to-peer protocol is employed to accelerate large file delivery to wide-scale target computers, by sharing their upload capacity in order to alleviate the bandwidth bottleneck of provisioning server. The previous study [17] presents a rapid provisioning mechanism of virtual containers for heterogeneous virtual machines. This study explains the cloud Infrastructure-as-a Service (IaaS) fulfillment procedure, the management of virtual machine kernel pool, and activation mechanism of multi-threading for rapid provisioning.

In the local cluster system, the network topology is open and rarely changed. Hence, previous studies [18–22] in the local cluster system consider the network characteristics as much as possible when designing for fast booting. The previous study [18] presents a deployment mechanism, called SonD, in which the writable snapshot technique is used in order to achieve fast service creation, a dynamic service mapping, and a network boot mechanism. The previous study [19] exploits the InfiniBand protocol offered by Nvidia corporation, which dramatically reduces latency and CPU overhead with the Remote Direct Memory Access (RDMA) technology. The RDMA technology bypasses the OS kernel with direct memory-to-memory transfers. The previous study [20] describes the simplified network booting method, called Grendel. This study exploits a robust PXE boot server, REST API, and node management in a binary file format. The previous study [21] presents an efficient maintenance technology for a cluster supercomputer, that is, a single shared root filesystem image which reduces management complexity and completely automates the process of bringing a new computer into the cluster. The previous study [22] introduces analysis results that usage of network mounted boot disks results in negligible run-time while achieving faster provisioning time.

A few previous studies [7–9] addressed that topology-aware deployment mechanism can accelerate the performance of deployment procedure. The previous study [7] compares delays of topology-aware routing and topology-unaware routing in a switch-based network infrastructure. The previous study [8] provides a minimal-contention multicasting algorithm from a single sender to multiple receivers in switch-based fat-tree configurations. The previous study [9] designs a peer-to-peer protocol that selects a peer computer with consideration of network distance proximity and transmission rate in Internet network infrastructure. The previous study [23] analyzes the lower bound and the upper bound of broadcasting operation for the general fat-tree configuration [24] in which branches nearer the top of the hierarchy are fatter or thicker than branches further down the hierarchy in order to maximize bandwidth and provide multiple redundant paths. The previous studies [25,26] provide simulation frameworks that are able to compare the performance of various fat-tree network configurations with different communication patterns and

computation-to-communication ratios. However, all of these studies [7–9,23–26] did not consider the optimal deployment mechanism in a given fat-tree network configuration. Compared to the previous studies [7–9,23–26], the problem of finding an optimal deployment mechanism in a *specific* fat-tree network is handled in this paper, while the optimal deployment mechanism in the *general* fat-tree network remains an open problem.

Recently, in wireless environments of Internet-of-Things (IoT), many studies [27,28] have shown progress in energy-efficient multipath transmission or management of multipath fading channels. They focus on efficient utilization of frequency modulation, which is beyond scope of this study.

3. Proposed Topology-Optimal Deployment

In this paper, we describe a method for minimizing the time required to distribute an operating systems file of approximately 5.1 Gbytes in size to a cluster supercomputer, configured with a fat-tree structure. Section 3.1 describes the detailed connection structure of the target supercomputer. Section 3.2 describes the baseline unicast-based distribution method of operating systems file currently used on the target supercomputer. Section 3.3 describes a minimal-time distribution method of operating systems files optimized for the target supercomputer architecture based on peer-to-peer technology.

3.1. System Model

The fifth supercomputer operated by the Korea Institute of Science and Technology Information (KISTI), named as *Nurion* [29], has a system configuration as shown in Figure 1. It consists of 8437 nodes, including 8305 many-core KNL CPUs and 132 CPU-only nodes. The 8437 nodes are connected in a fat-tree configuration using eight Intel switches with 768 ports and 264 Intel switches with 48 ports, providing a high-performance 100 G interconnect. The other 13 48-port switches are used to connect the burst buffers, and I/O devices.

Figure 2 shows the fat-tree configuration of the Nurion supercomputer using 8 768-port switches and 264 48-port switches. The fat-tree structure has a depth of four, where the deploy server is located at Layer 1, the 8 768-port core switches are located at Layer 2, the 264 48-port edge switches are located at Layer 3, and the compute nodes and relay servers are located at Layer 4. The deploy server at Layer 1 and the 8 768-port switches at Layer 2 are connected by dual links for fault tolerance, and each 768-port switch at Layer 2 and the 264 48-port switches at Layer 3 are also connected by dual links. Each 48-port switch at Layer 3 is connected to the compute nodes or relay servers at Layer 4 by a single link. Even if it is configured as dual links, only one link is always used. In other words, the deploy server in Layer 1 has eight links in the downward direction, each core switch in Layer 2 has one link in the upward direction, and 264 links in the downward direction. Each edge switch in Layer 3 has eight links in the upward direction and 32 links in the downward direction.

Although each edge switch numbered as 260 and 264 are connected to 32 relay servers in Figure 2, the edge switch numbered as 260 is connected to 15 relay servers and 17 end nodes in the real Nurion supercomputer. Also, the edge switch numbered as 264 is connected to 21 relay servers and 11 IO devices in the real Nurion supercomputer. However, for the sake of simplicity, this paper assumes that edge switches numbered from 1 to 259 are all connected to only end nodes, and edge switches numbered from 260 to 264 are all connected to only relay servers. In other words, it is assumed that 8288, not 8305, end nodes are connected to edge switches numbered from 1 to 259, and 160, not 132, relay servers are connected to edge switches numbered from 260 to 264. Note that this assumption requires a minor modification when applying the proposed scheme to the real

Nurion supercomputer, but it does affect the performance of the proposed scheme. In each edge switch, at most eight relay servers participate in the proposed scheme because the number of upward links in each edge switch is 8. In the edge switch numbered as 260 of the real Nurion supercomputer, only eight relay servers participate in the proposed method and the remaining seven relay servers can perform separately the file deployment for the 17 end nodes. The deployment from the remaining seven relay servers to the 17 end nodes within a single switch is simpler and faster than the deployment of the eight relay servers participating in the proposed scheme. In the edge switch numbered as 264 of the real Nurion supercomputer, only eight relay servers participate in the proposed method and the remaining 13 relay servers do not play a role of retransmission (in the edge switches numbered as 261, 262 and 263, only eight relay servers participate in the proposed method and the remaining 24 relay servers do not play a role of retransmission).

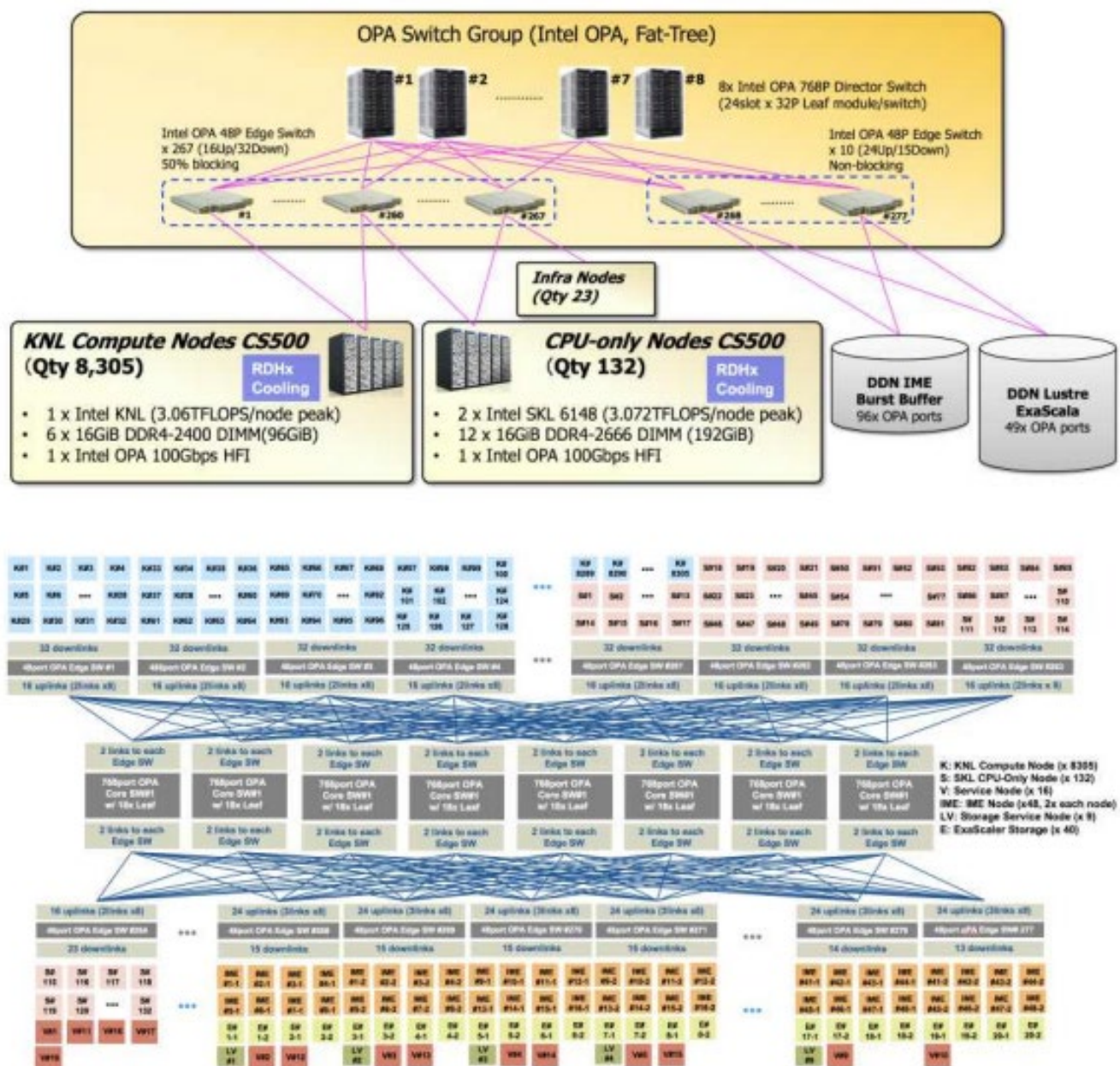


Figure 1. System configuration of Nurion supercomputer.

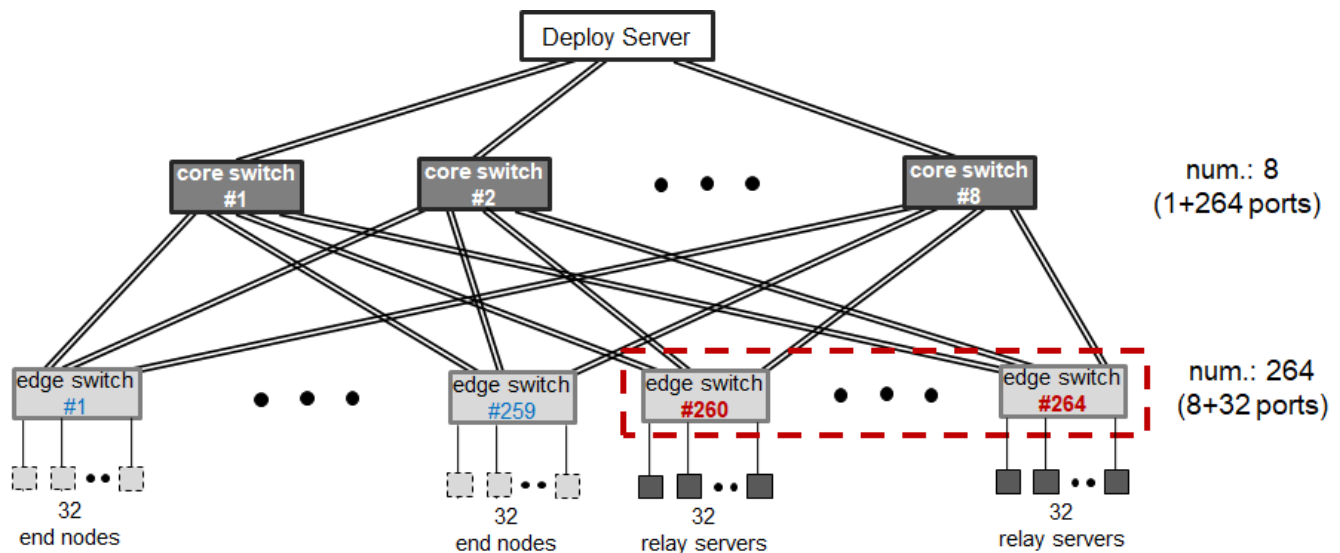


Figure 2. Switch-based fat-tree interconnection.

The commercial Bright Cluster Manager (BCM), named deploy server in this paper, deploys a new operating systems (OS) file of approximately 5.1 GB during each regular maintenance. The OS deployment process, as shown in Figure 3, begins with the BCM deploy server distributing the OS files to the target disk-based relay servers. Each relay server then transfers the OS file to the remaining diskless end nodes. The OPA connection depicted in Figure 3 represents the communication path via the core switch at Layer 2 and the edge switch at Layer 3 shown in Figure 2. Since the proposed method is designed to be specialized for the specific structure, the architectural characteristics such as uniform fat-tree structure, the number of upward links and downward links in switches, and locations of the deploy server, relay servers and end nodes are critical for the method.

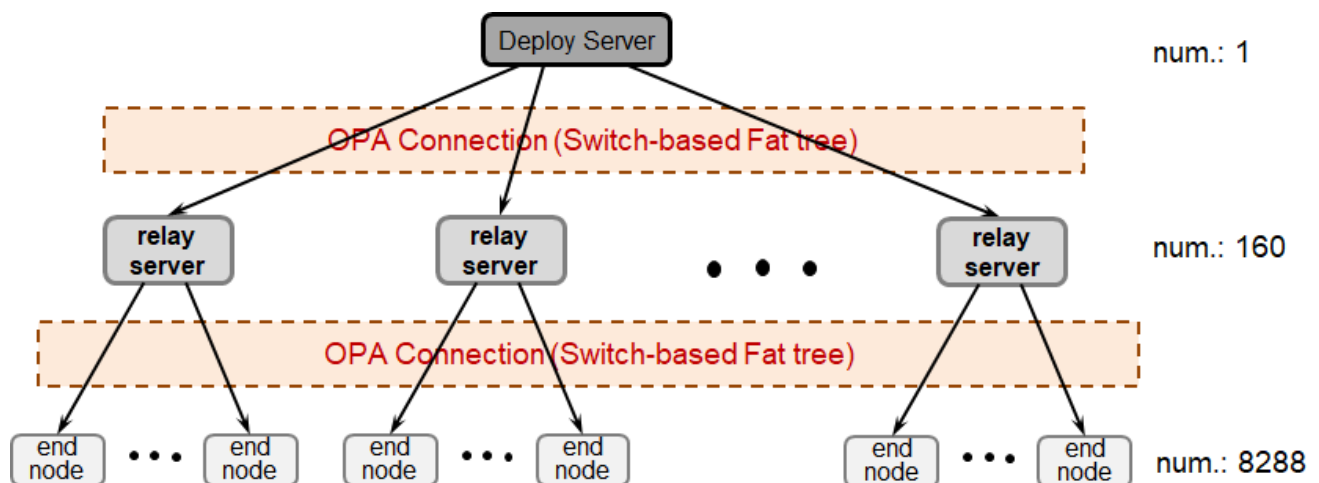


Figure 3. Outline of deployment procedure in Nurion supercomputer.

3.2. Baseline Deployment Method

Currently, on Nurion supercomputer, the deploy server performs only the transmission function, and the end nodes perform only the reception function, while the relay server performs both reception and transmission functions. Henceforth, the procedure for the deploy server to transmit the OS file to relay servers is referred to as Phase 1, and the procedure for relay servers to retransmit the received OS file to end nodes is referred to as Phase 2.

Figure 4a shows the Phase 1 procedure in which the deploy server delivers the OS file to 160 relay servers connected to edge switches numbered from 260 to 264. Figure 4b shows the Phase 2 procedure in which the 160 relay servers connected to edge switches numbered from 260 to 264 deliver the OS file to 8288 end nodes connected to edge switches numbered from 1 to 259. In Phase 1 of Figure 4a, the maximum number of paths that can be concurrently transmitted from the deploy server to the 160 relay servers is 8, which is determined by the number of core switches. In Phase 2 of Figure 4b, the maximum number of paths that can be concurrently transmitted from the relay servers to the end nodes is 40, which is obtained by summation of eight uplinks of the five edge switches numbered from 260 to 264. The maximum number of paths that can be transmitted from senders to receivers can be obtained automatically by applying the max-flow min-cut algorithm [30].

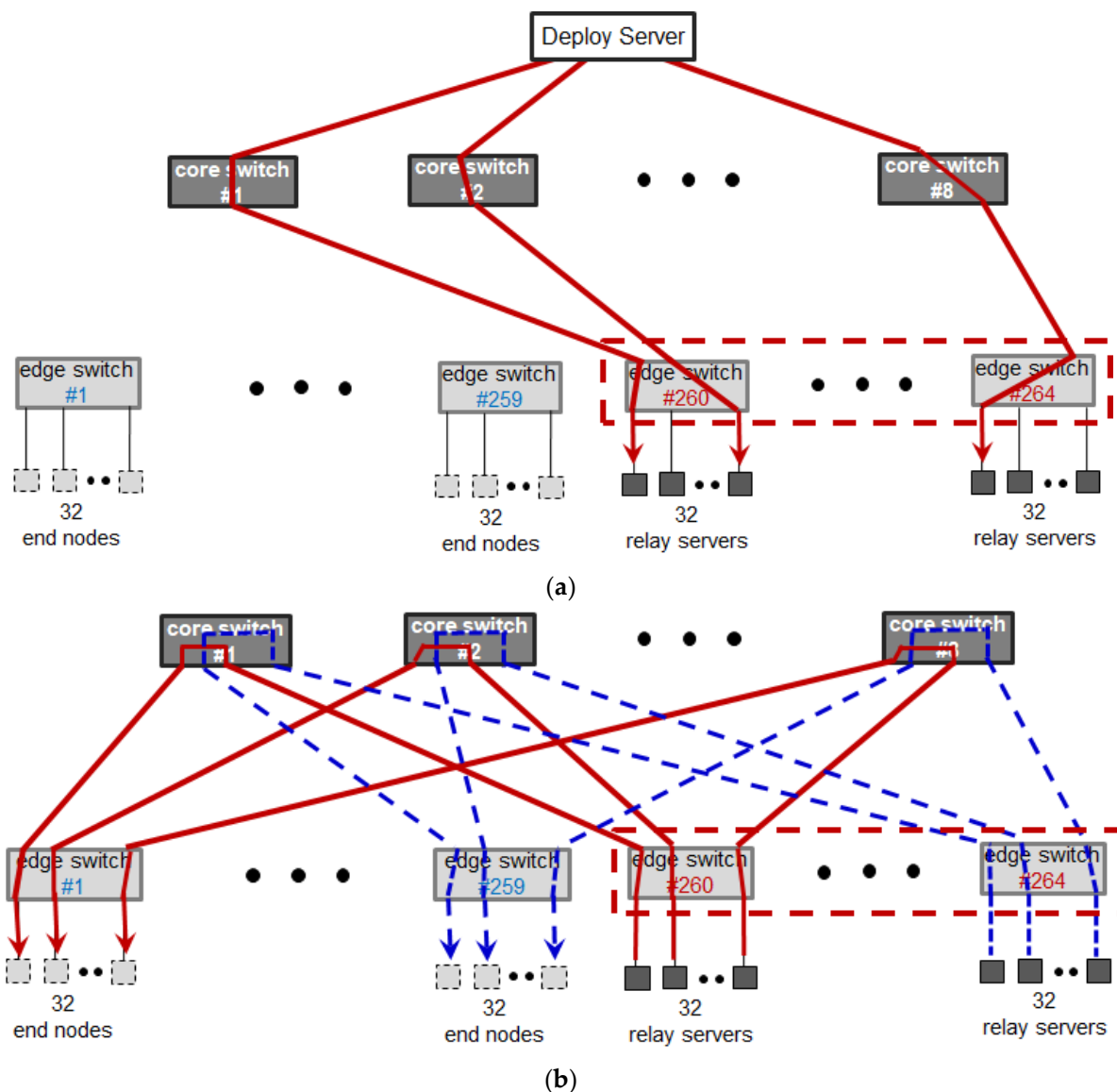


Figure 4. Baseline deployment method: (a) Phase 1 and (b) Phase 2.

In Phase 1 of Figure 4a, since the maximum number of paths that can be transmitted concurrently is 8, at least 20 ($=160/8$) transmission steps are required to transmit the operating systems file to 160 relay servers. In Phase 2 of Figure 4b, since the maximum number of paths that can be transmitted concurrently is 40, at least 207.2 ($=8288/40$) transmission steps are required to transmit the operating systems file to 8288 end nodes. In

other words, if the time required to transmit the operating systems file of approximately 5.1 Gbytes once is x , at least $20 \times x$ time is required in Phase 1, and at least $208 \times x$ time is required in Phase 2.

3.3. Proposed Deployment Method

In the baseline method described in Section 3.2, only the relay servers perform both receiving and transmitting functions. In the proposed method, all nodes perform both receiving and transmitting functions.

Figure 5a shows the number of increments of receiving nodes when the end node used in the baseline method performs only the receiving function. Since the number of receiving nodes increases by one at each step, the number of receiving nodes becomes n where n denotes the number of repeated communication steps. Figure 5b shows the number of increments of transmitting nodes when the end node used in the proposed method uses P2P technology to perform both receiving and transmitting. The number of receiving nodes becomes $2^0 + 2^1 + 2^2 + \dots + 2^{(n-1)} = (2^n - 1)$ for n communication steps. Figure 5c shows the increase in the number of receiving nodes when only the root node can transmit multiple communications simultaneously, while the remaining nodes can transmit only one communication at a time. Where k denotes the number of communications simultaneously transmitted by the root node, the number of receiving nodes for n communication steps becomes

$$k \times 2^0 + k \times 2^1 + k \times 2^2 + \dots + k \times 2^{(n-1)} = k \times (2^n - 1). \quad (1)$$

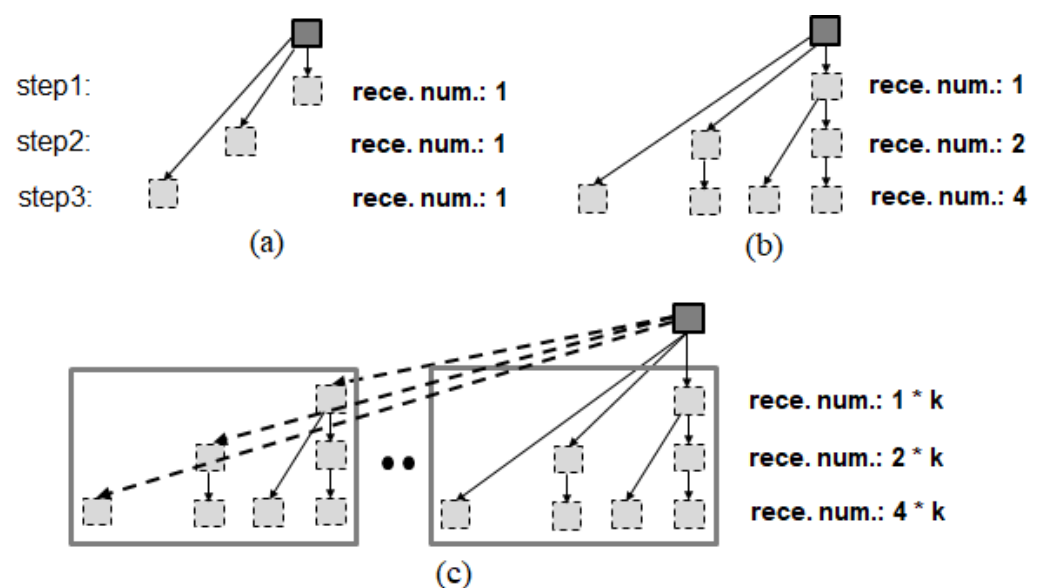


Figure 5. Expanding comparison of pure unicast and P2P transmission: (a) Expanding of pure unicast, (b) expanding of peer-to-peer unicast, and (c) expanding of peer-to-peer unicast with concurrent transmissions of root node.

Table 1 summarizes the minimum number of transmission steps required for the current baseline and the P2P-based methods. In Phase 1 of the baseline method, at least 20 ($=160/8$) transmission steps are required to deliver to 160 relay servers where the maximum number of concurrently transmittable paths is 8. In Phase 2, at least 208 ($>207.2 = 8288/40$) transmission steps are required to deliver to 8288 end nodes, where the maximum number of concurrently transmittable paths is 40.

Table 1. Summary of the required transmission steps.

	Num. of Minimum Steps in the Baseline Method	Num. of Minimum Steps in the P2P-Based Method
Phase 1	20	5
Phase 2	208	8

In the P2P-based method, the number of receiving nodes is expanded recursively in the manner illustrated in Figure 5c. As illustrated in Figure 4a, in Phase 1 of the OS deployment, the deploy server can concurrently transmit eight communications to relay servers each time, where $k = 8$. If the relay server also participates in the transmission task after receiving, the number of receiving nodes can be increased to a maximum of $8 \times (2^n - 1)$ according to Equation (1) by repeating the communication step n times. Then the minimum value of n satisfying the condition of $160 < 8 \times (2^n - 1)$ is 5 because $8 \times (2^4 - 1) = 120 < 160 < 8 \times (2^5 - 1) = 248$. As explained in Figure 4b, in Phase 2, the relay servers can concurrently transmit 40 communications to end nodes each time, where $k = 8$. In Phase 2 of the P2P-based method, the number of receiving nodes can be up to $40 \times (2^n - 1)$ according to Equation (1) by repeating the communication step n times. Then the minimum value of n satisfying the condition of $8288 < 40 \times (2^n - 1)$ is 8 because $40 \times (2^7 - 1) = 5080 < 8288 < 40 \times (2^8 - 1) = 10,200$. The deployment can be completed with the minimum number of transmission steps, only when the maximal paths concurrently transmittable (8 in Phase 1 and 40 in Phase 2) are distributed evenly to senders without contention between concurrent transmissions.

Figure 6 illustrates the broadcast procedure to 32 child nodes connected to a single edge switch used in the proposed method. Figure 6a shows the proposed method used in Phase 1 of the OS deployment, and Figure 6b shows the method used in Phase 2. In Figure 6, t_i represents the transmission at time step i . For clarity, Figure 6a omits the final t_5 , which transmits from odd to even nodes, and Figure 6b omits the final t_6 , which transmits from odd to even nodes. Figure 6a is used when the maximum number of concurrent transmissions is greater than the number of edge switches connected to target receiving nodes. Figure 6b is used when the maximum number of concurrent transmissions is much less than the number of edge switches connected to target receiving nodes. As shown in Figure 4a, in Phase 1, the maximum number of concurrent transmissions is 8 and the number of edge switches connected to the receiving relay servers is 5. And as shown in Figure 4b, in Phase 2, the maximum number of concurrent transmissions is 40 and the number of edge switches connected to target receiving end nodes is 259.

Figure 6a used in Phase 1 of the proposed method shows that if the OS file is transmitted concurrently to nodes 1 and 31 at time t_1 (or time t_2), the broadcasting procedure to 32 relay servers can be completed at time t_5 . Table 2 shows the cases of two simultaneous transmissions and a single transmission according to the time step to edge switches 260, 261, 262, 263, and 264 that connect the relay servers in the proposed method. In the transmission step t_1 , simultaneous transmissions are performed to each node 1 and each node 31 of edge switches 260, 261, and 262, and a single transmission is performed to each node 1 of the remaining edge switches 263 and 264. In the transmission step t_2 , a single transmission is performed to each node 17 of edge switches 260, 261, and 262, and simultaneous transmissions are performed to each node 17 and each node 31 of the remaining edge switches 263 and 264. Afterwards, a single transmission is performed to each node 25 of edge switches 260, ..., 265 at time t_3 , and a single transmission is performed to each node 29 of the five edge switches at time t_4 . At time t_5 , each odd node performs transmission to its right-side even node without transmission from the deploy server.

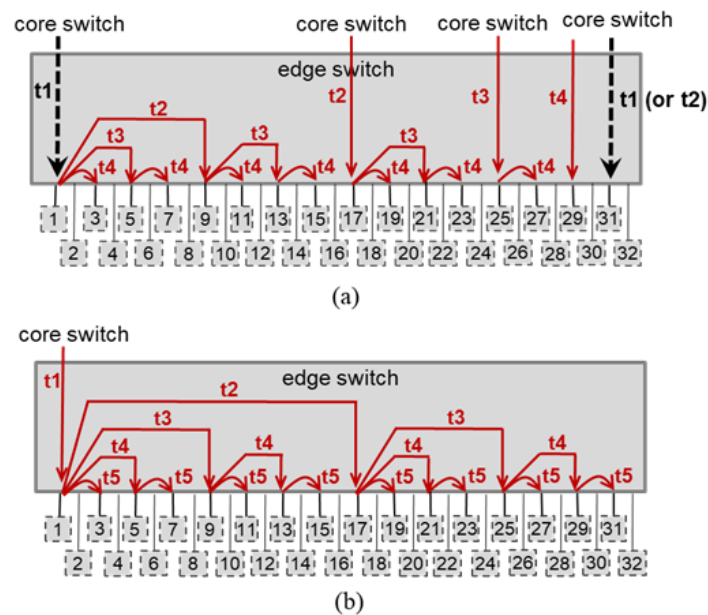


Figure 6. Broadcasting of 32 computing nodes in an edge switch: (a) 5-step multicast with concurrent transfers from core switches (not depicting the last step t_5) and (b) 6-step multicast with a single transfer from core switches (not depicting the last step t_6).

Table 2. Detailed operations in Phase 1 of the proposed method.

Time Step	Two Simultaneous Trans. from Deploy Server	Single Trans. from Deploy Server
t_1	edge switch 260, 261, 262	edge switch 263, 264
t_2	edge switch 263, 264	edge switch 260, 261, 262
t_3	none	edge switch 260, ..., 264
t_4	none	edge switch 260, ..., 264
t_5	none	none

Figure 6b used in Phase 2 of the proposed method shows that the broadcasting procedure to 8288 end nodes can be completed at time t_6 by consuming an additional five steps if each node 1 of target edge switches receives the OS file. In other words, if the OS file can be transmitted to each node 1 of edge switches numbered as 1, ..., 259 connecting target end nodes within m steps, the OS file transmission to a total of 8288 end nodes can be completed within $(m + 5)$ steps.

Figure 7 shows the procedure to complete transmission to each node 1 of edge switches numbered as 1, ..., 259 within three steps, which is the case of $m = 3$. According to Equation (1), the transmission to 259 edge switches can be completed in only three steps because $40 \times (2^3 - 1) = 280$, where the maximum number of concurrently transmittable paths is 40 through the eight upward links of edge switches 260, 261, 262, 263, and 264 for each step. In Figure 7, only 36 concurrent transmissions among the maximum 40 concurrent transmissions are used because the number 258 of target edge switches is less than 280. The covered areas that the node 1 of the edge switch 260 handles through three steps are the edge switches 1, ..., 7. The covered areas that the node 2 of the edge switch 260 handles through three steps are the edge switches 8, ..., 14. Similarly, the covered areas that the node 4 of the edge switch 264 handles through three steps are the edge switches 253, ..., 259. As illustrated in Figure 7, the transmission to each node 1 of 259 edge switches can be completed within three steps. As illustrated in Figure 6b, the transmission to the remaining 31 end nodes of the 259 edge switches can be completed in an additional five steps. Then,

the OS file transmission to 8288 end nodes can be completed in eight steps if there is no delay due to collision among concurrent P2P communications.

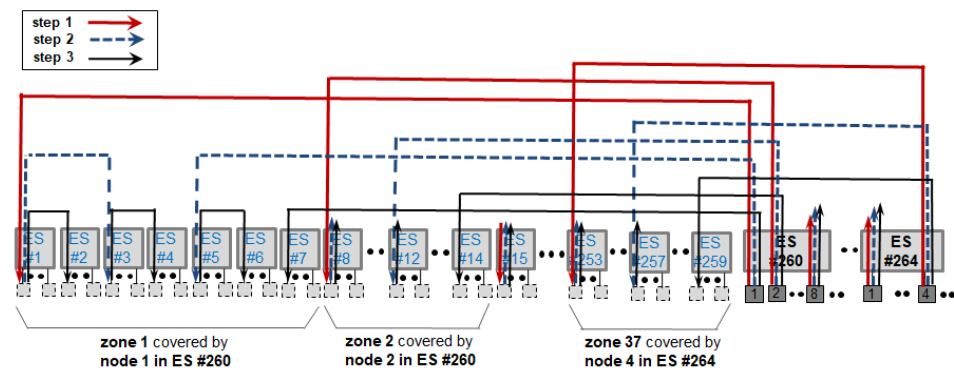


Figure 7. Multicast of 259 edge switches in Phase 1 of the proposed method.

Figure 8 shows that there is no overlapping in communication paths during the maximum 40 concurrent transmissions from relay servers to end nodes in Phase 2. Figure 8a shows all communication paths that occur simultaneously in step 1 of Phase 2. In the upward communication paths, the eight upward links from the edge switches 260, 261, 262, 263, and 264 are connected to eight core switches through different port numbers of each core switch. In the downward communication paths, each core switch is connected to 34 edge switches numbered as 1, 8, ..., 253 through different port numbers of each core switch. Therefore, there is no overlapping in step 1 of Phase 2. Figure 8b shows all communication paths that occur simultaneously in steps 2 and 3 of Phase 2. We can confirm that there is no overlapping of communication paths in the upward and downward directions in step 2. Similarly, there is no overlapping of communication paths in the upward and downward directions in step 3. Finally, concurrent transmissions within the edge switch shown in Figure 6b do not cause communication collision because most switch devices provide a multiplex function in the form of a full connection.

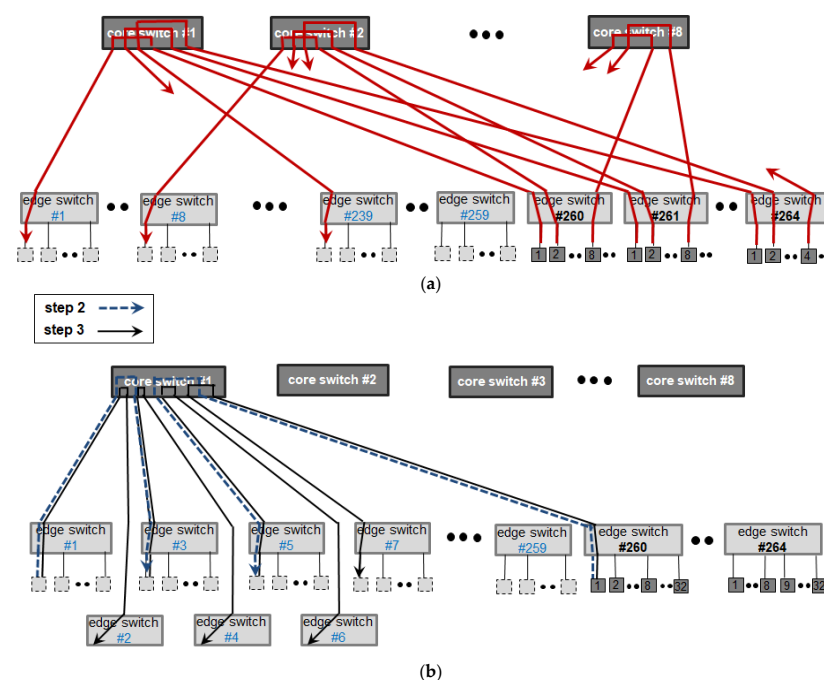


Figure 8. Contention-free routing path in Phase 2 of the proposed method: (a) All routing paths in step 1 and (b) all routing paths in step 2 and 3.

4. Evaluation

To evaluate the performance of the proposed method, we conducted numerical analysis and simulation experiments. Section 4.1 verifies through numerical analysis that the proposed method consumes the same number of steps as the optimal method. Section 4.2 shows through simulation experiments that the proposed P2P-based method described in Section 3.3 completes the OS deployment procedure much faster than the baseline method described in Section 3.2.

4.1. Numerical Analysis

In a P2P-based distribution method, in order to deliver to 160 relay servers in Phase 1 when the maximum number of simultaneously deliverable paths is 8, at least five transmission steps are required because $8 \times (2^4 - 1) = 120 < 160 < 8 \times (2^5 - 1) = 248$. In order to deliver to 8288 end nodes in Phase 2 when the maximum number of simultaneously deliverable paths is 40, at least eight transmission steps are required because $40 \times (2^7 - 1) = 5080 < 8288 < 40 \times (2^8 - 1) = 10,200$. That is, any distribution method based on the P2P technology requires at least five steps in Phase 1 and at least eight steps in Phase 2.

As explained in Section 3.3, the proposed method uses five transmission steps in Phase 1 and eight transmission steps in Phase 2. In addition, it is confirmed that no contention delay occurred due to overlapping routing paths among concurrent transmissions in Phase 1 and in Phase 2 either. That is, the proposed method is proven to be a method that minimizes the OS distribution time by using only the same number of transmission steps as the minimum number required by the P2P-based distribution method. The required communication steps of the optimal deployment method and the proposed method are summarized in Table 3. Note that the optimality analysis is applicable only to the minimum number of P2P unicast steps, not to wall-clock time in a fully realistic environment.

Table 3. Time comparison of the optimal method and the proposed method.

	Num. of Minimum Steps in the P2P-Based Method	Num. of Consumed Steps in the Proposed Method
Phase 1	5	5
Phase 2	8	8

4.2. Simulation Experiments

In order to compare the performance difference between the proposed deployment method and the existing baseline method described in Section 3.2, we implemented the two methods in using the Network Simulator 3 (NS-3) library [31] and Python programming language with version 3.13.9. End nodes play roles of both receiver and sender in the proposed method where they play a role of only receiver in the baseline method. The two methods find the maximal paths concurrently transmittable and distribute them evenly to the senders. The same 4-layer fat-tree structure as shown in Figure 2 was constructed.

Table 4 shows setting parameters used in the simulation experiments. Because the previous study [8] shows that the difference in transmission time depending on the number of traversed switches is negligible for large files, it is assumed that the number of switches passed through during transmission does not affect the transmission time of τ . The time τ was set to 11.5 s, which is measured under the TCP protocol in our previous experimental study [32]. The value α denotes the time delay occurring when a receiving node prepares the role of retransmitting in the next step. The value β denotes the time delay occurring when multiple communications are performed simultaneously within a switch device.

Table 4. Parameters of simulation experiments.

Notation	Parameter Role
τ	transmission time of single peer-to-peer unicast
α	overhead ratio of message forward
β	overhead ratio of switch multiplex

The baseline deployment method requires at least 228 transmission steps for both Phase 1 and Phase 2, while the proposed method requires 13 transmission steps as shown in Table 1. Therefore, the proposed method is approximately 17 ($\simeq 228/13$) times faster than the baseline method when calculating without the overhead incurred in the P2P-based method, where $\alpha = 0$ and $\beta = 0$.

We first examine the performance impact of overhead ratio α of message forward. The time overhead for retransmission preparation was reflected in the simulation performance evaluation, as shown in Figure 9. As the size of the OS file increases, the transmission time increases but the retransmission preparation time remains fixed. Hence, the ratio of retransmission preparation time to file transmission time decreases as the size of the OS file increases.

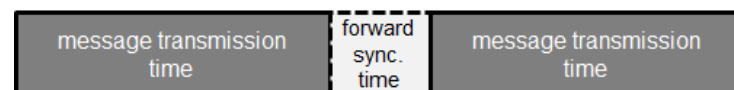
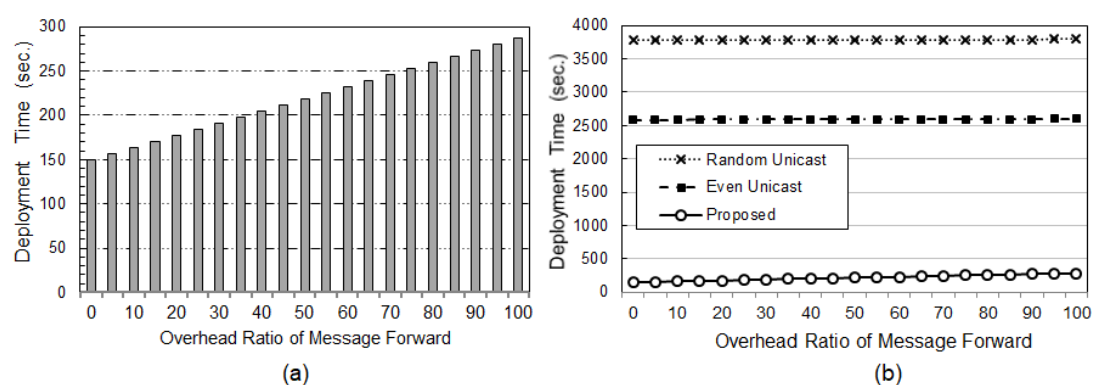
**Figure 9.** Forward sync. time.

Figure 10 shows experimental results against various overhead ratio of message forward. Figure 10a shows the deployment time of the proposed method. It consumes approximately 150 s when the message forward overhead is 0%, and approximately 288 s when the message forward overhead is 100%. Figure 10b shows the deployment times of the proposed method and two previous methods against various overhead ratio of message forward. ‘Even Unicast’ represents results of the baseline method described in Section 3.2, which operates based on the unicast and equally utilizes available paths. ‘Random Unicast’ represents results of a method that operates based on the unicast and randomly selects paths. As shown in Figure 10b, regardless of message forward overhead ratio, the proposed method is at least 9 times faster than ‘Even Unicast’ and at least 13 times faster than ‘Random Unicast’. The proposed method is about 25 times faster than the ‘Random Unicast’ method and about 13 times faster than the ‘Even Unicast’ method when overhead ratio is 0%. The proposed method is about 17 times faster than the ‘Random Unicast’ method and about 9 times faster than the ‘Even Unicast’ method when overhead ratio is 100%.

**Figure 10.** Deployment time against overhead ratio of message forward: (a) Deployment time of the proposed method and (b) comparison of deployment time between the proposed method and two previous methods.

Multiplexing operations, where multiple communications are performed simultaneously within a switch device, can incur delays due to processing operations compared to cases when only one communication is performed. We examine the performance impact of overhead ratio β of switch multiplex operations. Figure 11a shows the switch multiplex that occurs during the receiving process at the edge switch connecting relay servers in Phase 1 of OS file deployment process. Figure 11b shows the switch multiplex that occurs during the transmission process at the edge switch connecting relay servers in Phase 2. Figure 11c shows the switch multiplex that occurs within the core switch in Phase 2. Switch multiplex also occurs during the broadcast operations within a single switch, as shown in Figure 6. Switch multiplexing occurs not only in the proposed method but also in the previous methods.

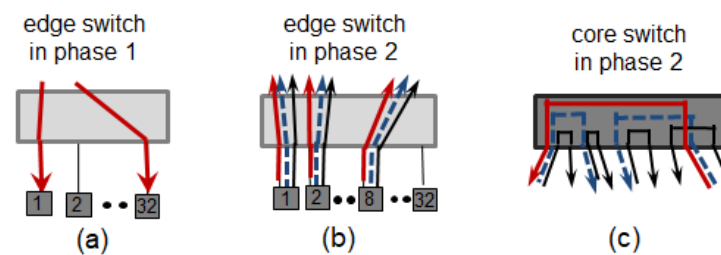


Figure 11. Cases of switch multiplex operations: (a) Edge switch connecting relay servers in Phase 1, (b) edge switch connecting relay server in Phase 2, and (c) core switch in Phase 2.

Figure 12 shows experimental results against various overhead ratio of switch multiplex. When two transmissions are performed simultaneously on a switch and the switch multiplex overhead value is set to $\beta\%$, a time delay of $\beta\%$ is added to the communication time. When k transmissions are performed simultaneously and the overhead value is set to $\beta\%$, a time delay of $(k - 1) * \beta\%$ is added to the communication time. Figure 12a shows the deployment time of the proposed method against various overhead ratios of switch multiplex. It consumes about 150 s when the switch multiplex overhead is 0%, and about 500 s when the overhead is 100%. Figure 12b the deployment times of the proposed method and the two previous methods against various overhead ratios of switch multiplex. As the switch multiplex overhead ratio increases, the increment in the deployment time of the previous ‘Even Unicast’ and ‘Random Unicast’ methods is significantly greater than that of the proposed method. The reason is that the proposed method requires 13 transmission steps, so the switch multiplex delay occurs within 13 steps at most. However, in the ‘Even Unicast’ method and ‘Random Unicast’ method, the switch multiplex delay occurs within 225 steps at most. Therefore, the increment in the deployment time due to the switch multiplex delay has a much greater impact on the previous methods than on the proposed method.

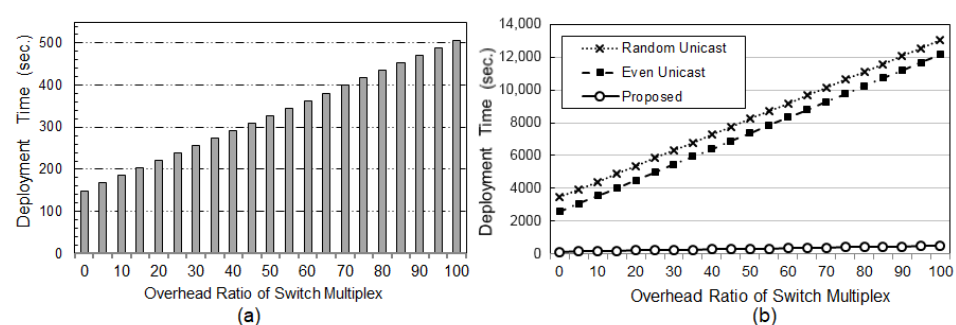


Figure 12. Deployment time against overhead ratio of switch multiplex: (a) Deployment time of the proposed method and (b) comparison of deployment time between the proposed method and two previous methods.

5. Conclusions and Discussion

In this paper, we propose an OS file deployment method optimized for the topological characteristics of a specific cluster supercomputer. The target topology is a fat tree, in which there are four layers and 8448 computing nodes are connected through 272 switches. The proposed method first finds the maximal paths concurrently transmittable from senders to receivers and next distributes the found maximal paths evenly to senders without contention between concurrent transmissions. Whereas previous deployment studies based on the peer-to-peer technology have reduced transmission delay of large files by partially considering topological characteristics, this study minimizes transmission delay by fully considering the topological characteristics. In this paper, we mathematically demonstrate that the proposed method is optimal such that it requires the minimum steps of peer-to-peer unicast for the deployment completion of OS file. Furthermore, simulation experiments show that the proposed method is at least nine times faster than the existing baseline methods.

The proposed method is designed to be optimal upon the idealized assumption that only the number of peer-to-peer unicast steps determines the overall deployment time. However, in real environments, its actual deployment time depends on many implementation details such as dynamics of network protocols, behaviors of CPU, disk and switches, actual link sharing, etc. So, the proposed method does not guarantee the minimum actual time in real environments. Also, if a node fails in the middle of retransmission procedure, the proposed method needs to find a substitute node replacing the failed node, which breaks the operational optimality of the proposed method; this remains a goal in our future study. Furthermore, the proposed method is optimized only for a specific cluster supercomputer architecture, called Nurion, and thus has the limitation of lower operational efficiency on other cluster supercomputers with different architectures. However, the two techniques introduced in this paper can be applicable with some modifications to other supercomputers with different architectures; the first is to find the maximal paths concurrently transmittable from senders to receivers and the second is to distribute the found paths evenly to senders without contention between concurrent peer-to-peer communications. To the best of our knowledge, this study is the first approach to minimize the number of peer-to-peer unicasts for the OS file deployment in a fat-tree topology. Our further study will investigate a deployment method with the minimum number of peer-to-peer recursive-expanding transmissions for the dragonfly topology [33], as well as the connection topology of the sixth supercomputer scheduled to be newly launched by the Korea Institute of Science and Technology Information (KISTI) in 2026.

Author Contributions: Conceptualization, J.R. and K.J.; methodology, W.Y.L.; formal analysis, W.Y.L.; data curation, S.R. and S.K.; Writing—Original Draft Preparation, W.Y.L.; Writing—Review and Editing, J.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Korea Institute of Science and Technology Information (KISTI). (No. K25L1M2C2).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

References

- Shiau, S.; Huang, Y.; Yen, C.; Tsai, Y.; Sun, C.; Juang, J.; Huang, C.; Huang, S. A Novel Massive Deployment Solution Based on Peer-to-Peer Protocol. *Appl. Sci.* **2019**, *9*, 296. [\[CrossRef\]](#)
- Dosanjh, M.; Bridges, P.; Kelly, S.; Laros, J. A Peer-to-Peer Architecture for Supporting Dynamic Shared Libraries in Large-Scale Systems. In Proceedings of the International Conference on Parallel Processing Workshop (ICPPW), Pittsburgh, PA, USA, 10–13 September 2012; Volume 41, pp. 55–61.
- Zhang, W.; Wu, Y.; Hur, N.; Ikeda, T.; Xia, P. FOBTv: Worldwide Efforts in Developing Next-Generation Broadcasting System. *IEEE Trans. Broadcast.* **2014**, *60*, 154–159. [\[CrossRef\]](#)
- Lee, K.; Teng, W.; Wu, J.; Huang, K.; Ko, Y.; Hou, T. Multicast and Customized Deployment of Large-scale Operating Systems. *Autom. Softw. Eng.* **2014**, *21*, 443–460. [\[CrossRef\]](#)
- Gronvall, B.; Marsh, I.; Pink, S. A Multicast-based Distributed File System for the Internet. In Proceedings of the ACM SIGOPS European Workshop: Systems Supports for Worldwide Applications, Connemara, Ireland, 9–11 September 1996; Volume 7, pp. 95–102.
- Androutsellis-Theotokis, S.; Spinellis, D. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Comput. Surv.* **2004**, *36*, 335–371. [\[CrossRef\]](#)
- Jeanvoine, E.; Sarzyniec, L.; Nussbaum, L. Kadeploy3: Efficient and Scalable Operating System Provisioning for HPC Clusters. *USENIX* **2013**, *38*, 38–44.
- Lee, W.Y.; Hong, S.; Kim, J. On Configuration of Switch-based Networks with Wormhole Switching. *J. Interconnect. Netw.* **2000**, *1*, 95–114. [\[CrossRef\]](#)
- Ren, S.; Tan, E.; Luo, T.; Chen, S.; Guo, L.; Zhang, X. TopBT: A Topology-Aware and Infrastructure-independent Bittorrent Client. In Proceedings of the IEEE INFOCOM, San Diego, CA, USA, 15–19 March 2010; pp. 1–7.
- Day, J.D.; Zimmermann, H. The OSI reference model. *Proc. IEEE* **1983**, *71*, 1334–1340. [\[CrossRef\]](#)
- ODonnell, C. Using BitTorrent to Distribute Virtual Machine Images for Classes. In Proceedings of the ACM SIGUCCS, Portland, OR, USA, 19–22 October 2008; Volume 36, pp. 287–290.
- Petrovic, S.; Brown, P. Large Scale Analysis of the eDonkey P2P File Sharing System. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 2746–2750.
- López-Fuentes, F.A.; García-Rodríguez, G. Collaborative Cloud Computing Based on P2P Networks. In Proceedings of the International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, Switzerland, 23–25 March 2016; Volume 30, pp. 209–213.
- Shiau, S.; Huang, Y.; Tsai, Y.; Sun, C.; Yen, C.; Huang, C. A BitTorrent Mechanism-Based Solution for Massive System Deployment. *IEEE Access* **2021**, *9*, 21043–21058. [\[CrossRef\]](#)
- Chen, Z.; Zhao, Y.; Lin, C.; Wang, Q. Accelerating Large-scale Data Distribution in Booming Internet: Effectiveness, Bottlenecks and Practices. *IEEE Trans. Consum. Electron.* **2009**, *55*, 518–526. [\[CrossRef\]](#)
- Chen, Z.; Zhao, Y.; Miao, X.; Chen, Y.; Wang, Q. Rapid Provisioning of Cloud Infrastructure Leveraging Peer-to-Peer Networks. In Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops, Montreal, QC, Canada, 22–26 June 2009; Volume 29, pp. 324–329.
- Liao, C.; Wu, C.; Young, H.; Chang, K.; Huang, H. A Novel mechanism for Rapid Provisioning Virtual Machines of Cloud Services. *IEEE Netw. Oper. Manag. Symp.* **2012**, *12*, 721–735.
- Yin, Y.; Liu, Z.; Tang, H.; Feng, S.; Jia, Y. SonD: A Fast Service Deployment System based on IP SAN. In Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–10.
- Dalessandro, D.; Wyckoff, P. Fast Scalable File Distribution over Infiniband. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Denver, CO, USA, 3–8 April 2005; p. 8.
- Bruno, A.E.; Guercio, S.J.; Sajdak, D.; Kew, T.; Jones, M.D. Grendel: Bare Metal Provisioning System for High Performance Computing. In Proceedings of the Practice and Experience in Advanced Research Computing, Portland, OR, USA, 26–30 July 2020; pp. 13–18.
- Daly, D.; Choi, J.H.; Moreira, J.; Waterland, A. Base Operating System Provisioning and Bringup for a Commercial Supercomputer. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Long Beach, CA, USA, 26–30 March 2007; pp. 1–7.
- Turk, A.; Gudimetla, R.; Kaynar, E.; Hennessey, J.; Tikale, S. An Experiment on Bare-Metal BigData Provisioning. In Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing, Denver, CO, USA, 20–21 June 2016; pp. 114–119.
- Bilardi, G.; Codenotti, B.; Del Corso, G.; Pinotti, C.; Resta, G. Broadcast and associative operations on fat-trees. In Proceedings of the Third International Euro-Par Conference on Parallel Processing, Passau, Germany, 26–29 August 1997; pp. 196–207.
- Lonare, A.; Gulhane, V. Addressing Agility and Improving Load Balance in Fat-tree Data Center Network—A Review. In Proceedings of the International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 26–27 February 2015; Volume 2, pp. 965–971.

25. Jain, N.; Bhatele, A.; Howell, L.; Böhme, D.; Karlin, I.; León, E.A. Predicting the Performance Impact of Different Fat-Tree Configurations. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 12–17 November 2017; pp. 1–13.
26. Liu, N.; Haider, A.; Jin, D.; Sun, X.-H. Modeling and Simulation of Extreme-Scale Fat-Tree Networks for HPC Systems and Data Centers. *ACM Trans. Model. Comput. Simul.* **2017**, *27*, 1–23. [[CrossRef](#)]
27. Gong, Y.; Zhang, Z.; Wang, K.; Gu, Y.; Wu, Y. IoT-Oriented Single-Transmitter Multiple-Receiver Wireless Charging Systems Using Hybrid Multi-Frequency Pulse Modulation. *IEEE Trans. Magnetics* **2024**, *60*, 8401606. [[CrossRef](#)]
28. Ma, H.; Tao, Y.; Fang, Y.; Chen, P.; Li, Y. Multi-Carrier Initial-Condition-Index-Aided DCSK Scheme: An Efficient Solution for Multipath Fading Channel. *IEEE Trans. Veh. Technol.* **2025**, *74*, 15743–15757. [[CrossRef](#)]
29. Nurion, 5th Sumpercomputer Summary. Available online: <https://www.ksc.re.kr/eng/resources/nurion> (accessed on 24 December 2025).
30. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2009.
31. Ivey, J.S.; Swenson, B.P.; Riley, G.F. Simulating Networks with NS-3 and Enhancing Realism with DCE. In Proceedings of the Winter Simulation Conference (WSC), Las Vegas, NV, USA, 3–6 December 2017; pp. 690–704.
32. Rho, S.; Ryu, S.; Jeong, G. A Study on the Deployment Strategy of Supercomputer Operating System Based on an Intelligent Cluster Management System. In Proceedings of the Fall Conference of the Korean Society for Internet Information, Taipei, Taiwan, 16–19 December 2024; Volume 25.
33. Kim, J.; Dally, W.J.; Scott, S.; Abts, D. Technology-driven, Highly-scalable Dragonfly Topology. In Proceedings of the International Symposium on Computer Architecture, Beijing, China, 21–25 June 2008; Volume 35, pp. 77–88.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.