PAPER

A Distributed Mechanism for Probing Overlay Path Bandwidth Using Local Information Exchange

Tien Hoang DINH[†a)], **Go HASEGAWA**[†], ***Members***, ***and*** **Masayuki MURATA**[†], ***Fellow***

SUMMARY Available bandwidth, along with latency and packet loss rate, is an essential metric for the efficient operation of overlay network applications. However, the measurement of available bandwidth creates a larger traffic overhead than other metrics. Measurement conflicts on route-overlapping paths can also seriously degrade measurement accuracy and cause a non-negligible increase in the network load. In this paper, we propose a distributed method for measuring the available bandwidth in overlay networks that can reduce measurement conflicts while maintaining high measurement accuracy at low cost. Our main idea is that neighboring overlay nodes exchange route information to detect overlapping paths and share the measurement results of overlapping paths to configure parameter settings for available bandwidth measurements. Our simulation results show that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. The measurement accuracy of our method remains better than that of the existing method when the total measurement traffic loads of both methods are equal.

key words: *overlay networks, network measurement, measurement conflict, distributed measurement method, information exchange*

1. Introduction

The estimation of available bandwidth is crucial for many overlay network applications. For example, available bandwidth information allows the construction of an efficient overlay topology for video on demand [1] and peer-assisted streaming [2].

However, in general, measuring available bandwidth in overlay networks is expensive, not only because of the huge amount of pair-wise measurements but also because of the large traffic load of each measurement. In particular, for an overlay network that contains n overlay nodes, the number of pair-wise measurements is $O(n^2)$, which is unacceptable in large-scale overlay networks. Furthermore, the traffic load of each measurement of the available bandwidth is much larger than that of measurement of other metrics, such as latency or packet loss rate. This is because latency or packet loss rate can be measured by such lightweight tools as `ping`, while measuring the available bandwidth requires more complicated and costly mechanisms. For example, for Pathload [3], [4], which is one of the most accurate tools for measuring end-to-end available bandwidth, groups of packet streams called packet fleets are sent at various rates within a large range that contains the real value of available bandwidth. The traffic load of one Pathload measurement is very large and can reach 10 MB, based on one study [5]. However, most existing solutions focus on decreasing the number of pair-wise measurements [6]–[10] rather than reducing the traffic load of each measurement.

Another measurement issue in overlay networks is measurement conflict, which degrades measurement accuracy. This problem occurs when measurement tasks of overlapping paths are performed simultaneously. Previous studies have addressed this problem, and algorithms for avoiding concurrent measurements of overlapping paths have been proposed [11]–[13]. However, in these methods, although measurement conflicts can be avoided completely, measurement frequency is small, which leads to inaccurate measurement results [14]. Furthermore, the concurrent measurements of overlapping paths do not always cause conflict, depending on the mechanism of the measurement tools. For example, in the case of Pathload, because the interval between two consecutive packet streams is set to a value not smaller than one RTT, if the sending time of one packet stream is smaller than one RTT, the probability that a conflict occurs is smaller than that of non-conflict.

In a previous study [14], we proposed a distributed method for measuring additive metrics such as latency or packet loss rate, that can reduce measurement conflict and improve measurement accuracy. In this method, overlay nodes exchange route information to detect overlapping paths. Based on the overlapping state, the measurement frequency and timing of each path are determined to reduce the measurement conflict. Overlay nodes then exchange the measurement results of the overlapping parts to improve the measurement accuracy of these parts and improve the measurement accuracy of the whole path. However, we cannot apply this method when the metric is bandwidth-related information such as available bandwidth or throughput, because the measurement results of the overlapping parts cannot be obtained.

In this paper, we propose a distributed method for measuring the available bandwidth that can also reduce the measurement conflict while decreasing the traffic load of each measurement. Even though we use the same mechanism as a previous study [14] to detect the overlapping paths, we introduce a novel method that determines the measurement frequencies and timings to better measure the available bandwidth. To obtain accurate measurement results, we adopt some mechanisms similar to induced-congestion-based end-to-end available bandwidth tools such as Pathload

Manuscript received August 8, 2013.
Manuscript revised December 31, 2013.

[†]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: d-hoang@ist.osaka-u.ac.jp

DOI: 10.1587/transcom.E97.B.981

or pathChirp [15] for measuring end-to-end available bandwidth. To reduce the measurement traffic load, the overlay nodes exchange the measurement results of the overlapping paths to calculate the parameters for each measurement.

We make the following contributions in this paper:

- We propose an algorithm that determines the measurement frequencies and timings of the overlapping paths to reduce measurement conflicts.
- We propose a method for calculating the parameters of each measurement to reduce the measurement traffic load.
- We evaluate our method and compare it with a previous method [12] by simulations with both generated and real Internet topologies.

From the simulation results, we reach the following conclusions:

- The relative errors in the measurement results of our method are approximately only 65% of those of the previous method [12].
- The measurement accuracy of our method is still better than that of the method in [12] when the total measurement traffic loads of both methods are equal.

The rest of this paper is organized as follows. Section 2 describes related work. Definitions related to overlay networks are presented in Sect. 3. In Sect. 4, we explain our method that reduces measurement conflicts while decreasing the traffic load of each measurement. We evaluate our method in Sect. 5 and conclude this paper in Sect. 6.

2. Related Work

Measuring end-to-end available bandwidth has been extensively studied, and many measurement tools have been proposed so far. These tools can be mainly divided into two categories: Probe Gap Model (PGM) tools, e.g., IGI [16] and Spruce [5], and Probe Rate Model (PRM) tools, e.g., Pathload and pathChirp. PGM tools set an initial time gap between consecutive probing packets at the sender and observe the changes of the time gaps at the receiver to infer the cross traffic rate. They then subtract the cross traffic rate from the physical capacity of the bottleneck link to obtain the available bandwidth. IGI finds an initial time gap that makes the queue of the bottleneck link full but not overflowed, while Spruce sets the initial time gap to the bottleneck link transmission time of a 1500 Byte packet. PRM tools rely on the idea of induced congestion: the delays of packets in a stream show an increasing trend when the probing rate exceeds the available bandwidth. These tools send packet streams at various rates within a large range, called *search range*, that contains the real value of available bandwidth, and try to determine the rate at which the delays of the probe packets start increasing. This rate is an estimation of the available bandwidth. Pathload uses periodic packet streams, while pathChirp exponentially increases the probing rate within one stream. A hybrid method that combines the algorithms of pathChirp and IGI is introduced in [17]. In general, PRM tools are more accurate but also more intrusive than PGM tools [18]. In this paper, to obtain accurate measurement results, we use PRM tools to measure the end-to-end available bandwidth. However, because the default configurations of these tools make them intrusive, we propose methods for configuring the search range of each measurement to reduce the measurement traffic load. The idea of setting the search range is also proposed in [17]. In contrast to our method, in which we try to estimate a search range that is near the real value of available bandwidth, the search range in [17] is generally large and does not approximate the real value of available bandwidth.

Many solutions have been proposed for effectively measuring the available bandwidth in overlay networks, and most focus on decreasing the number of pair-wise measurements from the $O(n^2)$ traffic load of full-mesh measurements [6]–[10]. The method in [6] selects and measures only some paths that cover all the links of the paths between the overlay nodes and bases on these results to roughly estimate the results of remaining paths. The measurement traffic load is reduced to $O(n \log n)$, but the accuracy of the measurement results obtained by this technique is not high. BRoute [7] relies on two characteristics of overlay networks constructed over the Internet: (1) bottleneck links exist from both ends of the overlay path in roughly four hops or less, and (2) path overlaps often exist near both ends of the overlay path. Therefore, the available bandwidth of a segment near both ends of each overlay path can be used to get the available bandwidth of the entire path, which greatly reduces the measurement traffic load. However, this technique requires BGP routing information in advance to infer the AS-level paths between end hosts. Currently proposed solutions [8]–[10] rely on the observation that the measurement of available bandwidth can be approximately embedded to metric spaces, and thus it can be estimated using the concept of distance in metric space. In [8], the available bandwidth between two arbitrary nodes is calculated based on the measurement results of the incoming and outgoing paths between these nodes and predetermined hosts called landmarks. Sequoia [9] embeds nodes in the leaves of a weighted tree and uses the distances in it to estimate the available bandwidth. Another method, which is a decentralized version of Sequoia, reduces the number of measurements [10]. Although these methods show better results than previous coordinate-based solutions [19], [20], their measurement accuracy remains insufficient because embedding the measurement of available bandwidth to a metric space is only approximately justified by some real Internet datasets. We take a contrasting approach to the existing solutions [6]–[10] and focus on decreasing the traffic load of each measurement. Our approach not only reduces the total measurement traffic load but also helps mitigate the measurement conflict.

The measurement accuracy can be evaluated by the relative error of measurement results, which is defined as follows. Assume that we must aggregate measurement results

of a path at a predetermined period of time. Denote $\bar{A}_{meas}$ as the average of the measurement results, and $\bar{A}$ as the average of the available bandwidth in that period. Then the relative error is defined as

$$e = \frac{|\bar{A}_{meas} - \bar{A}|}{\bar{A}}. \quad (1)$$

Because of the presence of cross traffic at multiple links of the path, the available bandwidth of the path fluctuates with time. In the case of large fluctuation of available bandwidth, the average value $\bar{A}_{meas}$ of a small number of measurement results, even if highly accurate, may be far different from $\bar{A}$, meaning that the measurement accuracy is low. In general, the more measurement results are obtained, the more $\bar{A}_{meas}$ gets closer to $\bar{A}$. Therefore, we must obtain as many accurate measurement results as possible to improve measurement accuracy. Since the accuracy of each measurement can be seriously affected by the conflicts between the concurrent measurements of overlapping paths [11], we should reduce the measurement conflict while maintaining high measurement frequency. However, existing solutions [11]–[13] focus on avoiding the measurement conflict by sacrificing measurement frequency. The main idea of these studies is that they use heuristic algorithms from graph theory to schedule the measurement timings of the paths so that the overlapping paths are measured at different timings. Although measurement conflicts can be completely avoided, the measurement frequencies are limited, and so the measurement accuracy is not high. Furthermore, the concurrent measurements of overlapping paths do not always cause measurement conflict; the probability that conflicts occur depends on the mechanism of the measurement tool and may be small in some cases. In particular, for such tools as Pathload or pathChirp [15], to obtain accurate results, many packet streams are redundantly sent, and the interval between packets is carefully calculated. Thus the number of packets that experience conflict may be so small that the measurement results are not affected. Our method does not completely avoid concurrent measurements of overlapping paths, as in previous solutions [11]–[13]; instead it reduces them while maintaining high measurement frequency to improve measurement accuracy. Furthermore, our method is a completely distributed method, in which each overlay node exchanges route information with neighbor nodes to detect path overlaps and determine measurement timings for the paths starting from itself. On the other hand, the methods in [11]–[13] are centralized methods, which use a controller to gather the route information of all paths and schedule the measurements for each path. Thus, the methods in [11]–[13] must deal with the problem of synchronizing the timers at all overlay nodes to start and stop measurements exactly at the timings scheduled by the controller. This is not the problem in our method, because each overlay node starts and stops measurements by itself.

In [14], we proposed a method for measuring such additive metrics as latency or packet loss rate that improve the measurement accuracy utilizing measurement exchanges between overlay nodes. The method contains three parts:

- an algorithm for detecting overlapping paths,
- an algorithm for determining the measurement timings that reduce measurement conflicts while maintaining high measurement frequency,
- and an algorithm for exchanging the measurement results to improve the measurement accuracy.

In this paper, we only use the same algorithm for detecting overlapping paths, but propose two new algorithms for determining the measurement timings and exchanging the measurement results. This is because the two algorithms in [14] can only be applied for additive metrics; they cannot be applied for such bandwidth-related information as available bandwidth or throughput for the following two reasons.

First, in [14], the source nodes of the overlapping paths exchange the measurement results of the overlapping parts to improve their measurement accuracy and consequently improve the measurement accuracy of the whole path. However, when the metric is bandwidth-related information, we cannot obtain the measurement results of the overlapping parts. In this paper, we propose a method for exchanging the measurement results of the whole path and the related information. Furthermore, because the measurement results of overlapping paths are equal only when these paths have the same bottleneck links, we cannot use such data to directly improve the measurement accuracy. We instead use them to configure parameters for each measurement to reduce the measurement time and traffic load. This not only reduces the total measurement traffic load but also mitigates measurement conflicts, indirectly improving the measurement accuracy.

Second, in [14], because the measurement results of one path can be used directly to improve the measurement accuracy of its overlapping paths, the number of measurements and measurement timings of each path can be roughly determined, as long as the total number of the measurements of these overlapping paths is maintained. However, in this paper, as explained above, because the measurement results of one path cannot be used directly for improving the measurement accuracy of its overlapping paths, the number of measurements and the measurement timings of each path must be determined more strictly. We thus propose a novel method for this end.

3. Network Model and Definitions

In this paper, we define an overlay network as a logical network constructed on an under-layer IP network. In overlay networks, overlay nodes are often installed on end hosts as an application program. In this case, both routing and traffic control at the overlay level are conducted at the end hosts, and such controls cannot be activated inside the network. On the other hand, the overlay routing inside the network becomes possible by installing overlay nodes on the routers in the network. This installation can be done in the networks that support configurations at the application level

in the routers. If the network supports such techniques as network virtualization [21] and software defined networks [22], which enable the settings of all network components at some devices called controllers, this installation can be further simplified. In this paper, to realize efficient routing control by overlay networks, we consider an overlay network in which the overlay nodes are deployed on both routers and end hosts.

We call an under-layer IP network an *underlay network* and consider an underlay network with m end hosts or routers, denoted by R_i $(i = 1, ..., m)$. For simplicity, we call an end host or router an *underlay node*. Suppose that n $(n \leq m)$ overlay nodes are deployed on n different underlay nodes. We denote the overlay nodes as O_i $(i = 1, ..., n)$. Density σ of the overlay nodes is defined as the ratio of the number of overlay nodes to the number of underlay nodes, i.e., $\sigma = n/m$. Figure 1 shows an example of an underlay network and the overlay network constructed on it. The gray arrows show the overlay paths, and the black arrows illustrate the underlay paths of the corresponding overlay paths. We assume the shortest path algorithm for routing in the underlay network and denote the underlay path between underlay nodes R_i and R_j as R_iR_j. For path R_iR_j, R_i is the *source node*, and R_j is the *destination node* of the path. If different paths R_iR_j and R_sR_t share at least one link, R_iR_j and R_sR_t overlap each other, or R_iR_j (R_sR_t) is an *overlapping path* of R_sR_t (R_iR_j). We define a *route* from R_i to R_j as a sequence of underlay nodes that construct an underlay path from R_i to R_j.

As in previous work [14], we classify the overlapping states into the following three types:

- Complete overlapping: One path completely includes another path. The path that includes the other is called the *longer path*, and the other path is called the *shorter path*.
- Half overlapping: Two paths share a route from the source node to a router that is not an overlay node.
- Partial overlapping: Two paths share a route that does not include the source node.

For example, in Fig. 1, path O_1O_3 is a complete overlapping path of O_1O_6. Paths O_1O_2 and O_1O_4 have a half overlapping relation. Path O_1O_4 is a partial overlapping path of O_2O_5.

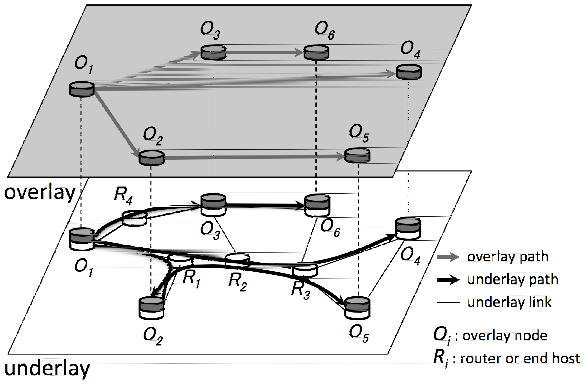


Fig. 1 Example of overlay network and path overlapping.

4. Proposed Method

4.1 Overview

Our solution is built in a completely distributed fashion, in which each overlay node measures the paths starting from itself, based on the information it exchanges with neighboring overlay nodes. The measurement procedure of each overlay node includes the following three phases:

- Detection phase of path overlaps
  The overlay nodes detect path overlaps using a previously described method [14].
- Calculation phase of measurement timings
  The measurement frequencies and timings in a predetermined duration are calculated based on the path-overlapping status.
- Measurement phase
  At each measurement timing in the predetermined duration, the overlay node calculates the parameters of the end-to-end measurement using the previous measurement results received from other nodes. The measurement is performed or omitted based on the calculation results. The overlay node then sends the measurement results and related information to the neighboring overlay nodes.

Figure 2 illustrates the relationships among the three phases. We call a duration that contains one calculation phase of measurement timings and one measurement phase an *aggregation period*. The aggregation period is determined by the user of the proposed method. It can be set to the period that the user requires the information of available bandwidth. Because the change in the routing information of the underlay network is generally less frequent than the change in the measurement results, the interval between two successive detection phases of path overlaps is larger than an aggregation period. We call this a *topology detection interval*. In general cases, the lengths of the detection phase of the path overlaps and the calculation phase of the measurement timings are much smaller than that of the measurement phase. This is because the overheads of the exchanging path information and the calculating measurement timings are very small compared to that of the measurements [14]. Therefore, we ignore the time of the detection phase of

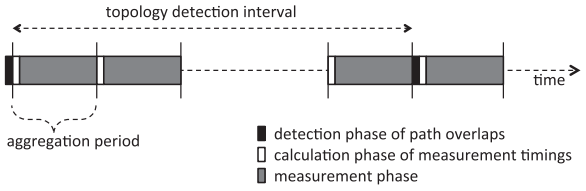


Fig. 2 Measurement procedure.

the path overlaps and the calculation phase of the measurement timings and only consider the time of the measurement phase when calculating the measurement frequencies.

4.2 Detection Phase of Path Overlaps

We use a previous method [14] for detecting complete, half, and partial overlapping paths on overlay networks. In particular, arbitrary overlay node O_i can detect complete and half overlapping paths of path O_iO_j by issuing `traceroute` to all the other nodes. To detect the partial overlapping paths of O_iO_j, O_i first utilizes the overlapping status of the half overlapping paths to find the candidates of partial overlapping paths and then exchanges the routing information with the source nodes of the candidates to determine their overlapping states. For example, in Fig. 1, we infer that path O_2O_5 is a partial overlapping path of O_1O_4, because the length of the overlapping part of O_1O_4 and O_1O_2 is smaller than the length of the overlapping part of O_1O_4 and O_1O_5. O_1 then exchanges routing information with O_2 to confirm whether O_2O_5 is actually a partial overlapping path of O_1O_4. Our simulation results show that our method can detect about 90% of the partial overlapping paths with relatively small overhead [14].

4.3 Calculation Phase of Measurement Timings

We propose a method for calculating the measurement timings of the paths that can reduce measurement conflicts while maintaining high frequencies to improve measurement accuracy. Our method utilizes the overlapping status of the paths.

For complete overlapping paths, to avoid measurement conflict, we only measure the shorter path; the longer path is not directly measured, and its measurement result is estimated based on the measurement results of the shorter paths included in it [14].

We explain the method for half and partial overlapping paths as follows. Consider path O_iO_j that has half and partial overlapping paths (Fig. 3). We denote the number of half overlapping paths of O_iO_j as $(G_{i,j} - 1)$ $(G_{i,j} \geq 1)$. For simplicity, we rewrite $G_{i,j}$ as G. We denote path O_iO_j as path 1, and each of its half overlapping paths as path p $(2 \leq p \leq G)$. We also assume that path p $(1 \leq p \leq G)$ has $(K_p - 1)$ partial overlapping paths $(K_p \geq 1)$.

Overlay node O_i can avoid the measurement conflicts among half overlapping paths 1, 2, ... and G simply by measuring them sequentially. On the other hand, because the source nodes of the partial overlapping paths of path p are different, measurement conflicts between them cannot be completely avoided. Therefore, we propose a technique that combines a sequential measurement for half overlapping paths and a random measurement for partial overlapping paths. We set the time required to measure a single path to a predetermined parameter τ. We divide each aggregation period into T $(T \geq 1)$ *measurement time slots*, whose length is τ. We denote the measurement times of path p at an aggregation period as h_p $(h_p \leq T)$ and calculate h_p as follows.

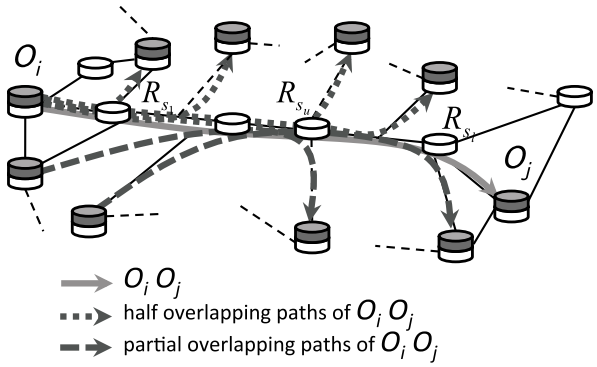


Fig. 3 Example for explaining the proposed measurement method.

We introduce β_p as a value that reflects the variability of the measurement results of path p at an aggregation period. Note that the method to determine β_p is beyond the scope of this paper. For example, β_p can be calculated based on the statistics of the measurement results or using a previous method [23]. We set measurement times h_p proportional to β_p for all paths, i.e., $h_1/\beta_1 = h_2/\beta_2 = ... = h_G/\beta_G$. To avoid measurement conflicts between half overlapping paths, the sum of their measurement times should be equal to or less than T: $\sum_{p=1}^{G} h_p \leq T$. We have $h_p \leq T\beta_p/(\sum_{s=1}^{G} \beta_s)$.

To reduce the measurement conflicts between path p and its $(K_p - 1)$ partial overlapping paths, we set the measurement times of path p to a value equal to or less than T/K_p, i.e., $h_p \leq T/K_p$. In addition, we keep the measurement times as large as possible to obtain as many measurement results as possible. Therefore, we set $h_p = min\{T\beta_p/(\sum_{s=1}^{G} \beta_s), T/K_p\}$.

Algorithm 1 Method for allocating measurement timings

```
1: function AllocMeasTime()
2: for p = 1 to G do
3:     Set c (c ≤ T) as the number of slots that have not been allocated to
       any path
4:     Divide these c slots into h_p groups, so that each group contains c/h_p
       continuous slots
5:     Randomly choose one slot from each group and allocate it for path p
6: end for
7: end function
```

However, the measurement times of path p should be large enough to avoid serious degradation of the measurement accuracy. To avoid this unexpected situation, we set the minimum of the measurement times to parameter D $(D \geq 2)$ and adjust h_p so that $h_p \geq D$, $1 \leq p \leq G$, and $\sum_{p=1}^{G} h_p \leq T$. In detail, for paths p where $h_p < D$, we set $h_p = D$. As a result, $\sum_{p=1}^{G} h_p$ increases and may exceed T. In

that case, for paths p where $h_p > D$, we reduce h_p one by one until $\sum_{p=1}^{G} h_p = T$ while keeping $h_p \geq D$. Note that if we set the aggregation period large enough, that is, when we set $T \geq D(n-1)$, because $G \leq n-1$, the above adjustment of h_p can be satisfied.

Next, we explain our method for randomly deciding the measurement timings of path p so that the measurement times of path p become h_p. The main idea of our method is that we divide T measurement time slots of an aggregation period into h_p groups and randomly choose one slot from each group to allocate for path p. Algorithm 1 shows the details of our method.

4.4 Measurement Phase

In this section, we explain our method that sets the parameters for each end-to-end measurement to reduce the measurement traffic load. We first present the method in the case that end-to-end measurement tool is Pathload [3], [4]. We then briefly explain the methods for other tools such as pathChirp [15].

4.4.1 Calculating Parameters for Available Bandwidth Measurement

To obtain accurate measurement results, we adopt a mechanism similar to Pathload for measuring the end-to-end available bandwidth. However, the default settings of parameters in each Pathload measurement makes its traffic load very large. Therefore, we propose a statistical method for calculating these parameters to reduce the measurement traffic load.

We first describe the mechanism of Pathload and explain why its measurement traffic load is large. Pathload relies on the fact that the one-way delays of a periodic packet stream show an increasing trend when the stream rate exceeds the available bandwidth. It first sets a large range (A_L, A_U), and uses a binary search algorithm to find the value of the available bandwidth inside this range. In detail, at each iteration of the measurement, the source node sends a string of packet streams called a *packet fleet* at the rate $A^* = (A_L + A_U)/2$ and checks whether there is an increasing trend in the one-way delays to judge if the real value of the available bandwidth is larger or smaller than the rate. If it found that the real value of the available bandwidth is larger than the rate, then it updates A_L to A^*, while in other case, it updates A_U to A^*, and repeats the search procedure. It stops when the width of the search range (A_L, A_U) is smaller than a predetermined threshold ω, and reports (A_L, A_U) as the measurement result. It is obvious that the traffic load of each measurement depends on the width of the initial search range. The initial value of A_L is set to 0, and that of A_U is set to a large value, for example, the capacity of the path; thus the measurement traffic load is very large [5].

To reduce the traffic load of each measurement, in our method, overlay nodes exchange measurement results of overlapping paths and related information to calculate a search range (A_L, A_U) that is narrow and near the real value of the available bandwidth. Our method relies on the following observations. First, because in general the available bandwidth of a path varies continuously, the probability that the most recent measurement results of the path is near the real value of available bandwidth is high. Therefore, we can use the most recent measurement results of a path to estimate the search range of the next measurement of that path. Second, because when the tight links of two overlapping paths belong to their overlapping part, their measurement results are equal, we can also use the most recent measurement results of overlapping paths to calculate the search range.

We explain our proposed method by describing the detailed behavior for path O_iO_j. We first assume that path O_iO_j has K partial overlapping paths ($K \geq 1$), denoted as $O_{u_s}O_{v_s}$ ($1 \leq s \leq K$). O_i receives the following information from O_{u_s} ($1 \leq s \leq K$).

1. Measurement result
2. Probability that a tight link of $O_{u_s}O_{v_s}$ belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$.

We denote the probability that the tight link of $O_{u_s}O_{v_s}$ belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$ as $\Phi_{O_{u_s}O_{v_s},O_iO_j}$ and calculate it using a previous method [23] as follows:

$$\Phi_{O_{u_s}O_{v_s},O_iO_j} = \frac{Latency(Overlap(O_iO_j, O_{u_s}O_{v_s}))}{Latency(O_{u_s}O_{v_s})}. \quad (2)$$

Here, $Overlap(O_iO_j, O_{u_s}O_{v_s})$ is the overlapping part of paths O_iO_j and $O_{u_s}O_{v_s}$.

After receiving the above data, O_i also estimates $\Phi_{O_iO_j,O_{u_s}O_{v_s}}$, which is the probability that the tight link of O_iO_j belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$. Then it calculates $\alpha_s = \Phi_{O_iO_j,O_{u_s}O_{v_s}}\Phi_{O_{u_s}O_{v_s},O_iO_j}$, which is the probability that the tight links of O_iO_j and $O_{u_s}O_{v_s}$ belong to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$. That means that α_s is the probability that the measurement results of O_iO_j and $O_{u_s}O_{v_s}$ are equal.

O_i stores its measurement results and the information received from other nodes. It uses the stored data to calculate A_L and A_U and discards them when it decides that these data are no longer useful for this calculation.

We assume that at measurement timing t^*, O_i stored G measurement results of O_iO_j and its half and partial overlapping paths, denoted as $(A_L^1, A_U^1), (A_L^2, A_U^2), ..., (A_L^G, A_U^G)$. We also denote the probabilities that these results equal the measurement results of O_iO_j as $\alpha_1, \alpha_2, ..., \alpha_G$, respectively. Note that α_s ($1 \leq s \leq G$) corresponding to the measurement result of O_iO_j is set to 1.

We calculate the lower bound of the 95% confidence interval of A_L^s ($1 \leq s \leq G$), denoted as S_L^*, and the upper bound of the 95% confidence interval of A_U^s ($1 \leq s \leq G$), denoted as S_U^*, as follows:

$$S_L^* = \bar{A}_L - 1.96\sqrt{\frac{V_L}{G}}, \quad S_U^* = \bar{A}_U + 1.96\sqrt{\frac{V_U}{G}}. \quad (3)$$

Here, $\bar{A}_L$, V_L, $\bar{A}_U$, and V_U are the weighted means and variances, calculated as follows:

$$\bar{A}_L = \sum_{s=1}^{G} \beta_s A_L^s, \quad V_L = \sum_{s=1}^{G} \beta_s {A_L^s}^2 - \bar{A}_L^2$$
$$\bar{A}_U = \sum_{s=1}^{G} \beta_s A_U^s, \quad V_U = \sum_{s=1}^{G} \beta_s {A_U^s}^2 - \bar{A}_U^2, \tag{4}$$

where $\beta_s = \alpha_s / \sum_{w=1}^{G} \alpha_w$ $(1 \le s \le G)$ is the weight of result (A_L^s, A_U^s).

When the stored data of O_i contain some measurement results of O_iO_j, we can infer that range (S_L^*, S_U^*) is near the real value of the available bandwidth and set $A_L = S_L^*$ and $A_U = S_U^*$. However, when they only contain measurement results of overlapping paths of O_iO_j, it is possible that these measurement results are much different from those of O_iO_j. As a result, the probability that range (S_L^*, S_U^*) is near the real value of the available bandwidth of O_iO_j is small and we can not use it as the search range. In such cases, we set $A_L = 0$ and $A_U = C^0_{O_iO_j}$, where $C^0_{O_iO_j}$ is the capacity of the IP link of path O_iO_j that connects to O_i.

4.4.2 Performing Measurement

Note that when the stored data of O_i contain some measurement results of O_iO_j, the width of range (S_L^*, S_U^*) reflects the variability of the available bandwidth of O_iO_j at the time near t^*. Therefore, O_i decides whether it will perform or omit the measurement of O_iO_j at measurement timing t^* based on this value as follows:

- If $S_U^* - S_L^* < \epsilon$, where ϵ is a predetermined parameter, we conclude that the variability of the available bandwidth of O_iO_j at the time near t^* is small and infer that this trend does not change at timing t^*. This means that (S_L^*, S_U^*) can be approximately considered as the measurement result at t^*. Therefore, O_i omits a measurement at timing t^* and uses (S_L^*, S_U^*) as the measurement result. This omission helps to decrease the measurement traffic load of O_i and reduce the conflicts between the measurements of O_iO_j and its partial overlapping paths.
- If $S_U^* - S_L^* \ge \epsilon$, O_i decides to perform a measurement at timing t^*.

Next, we explain how O_i performs a measurement when $S_U^* - S_L^* \ge \epsilon$. Although we infer that the real value of the available bandwidth is near range (S_L^*, S_U^*), we are not sure whether the real value of available bandwidth is inside this range. Therefore, O_i first sends probing packet streams with rates S_L^* and S_U^* to determine if the real value of the available bandwidth is between S_L^* and S_U^*, based on the status of the increasing trend in the one-way delays. If the real value of the available bandwidth does not exist between S_L^* and S_U^*, we infer that it has changed greatly and discard the stored measurement results because these data have become

Algorithm 2 Measurement algorithm for path O_iO_j

function MeasureOnePath()
//initialization
if stored data of O_i contain measurement results of O_iO_j **then**
  $A_L \leftarrow S_L^*$
  $A_U \leftarrow S_U^*$
else
  $A_L \leftarrow 0$
  $A_U \leftarrow C^0_{O_iO_j}$
end if
$ub\_found \leftarrow 0$
$lb\_found \leftarrow 0$
$meas\_time \leftarrow \tau$

//find a range (A_L, A_U) that includes available bandwidth
while $(ub\_found = 0 \,\|\, lb\_found = 0)$ && $(A_L > 0 \,\|\, A_U < C^0_{O_iO_j})$ && $meas\_time > 0$ **do**
  //test if A_U is an upper bound of the available bandwidth
  if $ub\_found = 0$ **then**
    $TestUB(A_L, A_U, lb\_found, ub\_found, meas\_time)$
  end if
  //test if A_L is a lower bound of the available bandwidth
  if $lb\_found = 0$ && $meas\_time > 0$ **then**
    $TestLB(A_L, A_U, lb\_found, ub\_found, meas\_time)$
  end if
end while

//measure available bandwidth between the range (A_L, A_U)
if $A_U - A_L > \omega$ && $meas\_time > 0$ **then**
  $RuntimeLimitedPathload(A_L, A_U, meas\_time)$
end if
return A_L, A_U
end function

Algorithm 3 Test of upper bound of search range

function TestUB($A_L, A_U, lb\_found, ub\_found, meas\_time$)
Set current time to $start\_time$
Send a packet fleet with rate A_U
if increasing trend **then**
  $ub\_found \leftarrow 1$
else
  $A_L \leftarrow A_U$
  $lb\_found \leftarrow 1$
  $A_U \leftarrow min(A_U + (S_U^* - S_L^*)/2, C^0_{O_iO_j})$
end if
Set current time to end_time
$meas\_time \leftarrow meas\_time - (end\_time - start\_time)$
return $A_L, A_U, lb\_found, ub\_found, meas\_time$
end function

Algorithm 4 Test of lower bound of search range

function TestLB($A_L, A_U, lb\_found, ub\_found, meas\_time$)
Set current time to $start\_time$
Send a packet fleet with rate A_L
if non increasing trend **then**
  $lb\_found \leftarrow 1$
else
  $A_U \leftarrow A_L$
  $ub\_found \leftarrow 1$
  $A_L \leftarrow max(A_L - (S_U^* - S_L^*)/2, 0)$
end if
Set current time to end_time
$meas\_time \leftarrow meas\_time - (end\_time - start\_time)$
return $A_L, A_U, lb\_found, ub\_found, meas\_time$
end function

unreliable. We also infer that its real value exists outside but near range (S^*_L, S^*_U). We then set a new search range at a neighboring range of (S^*_L, S^*_U) and check whether the real value of the available bandwidth belongs to this new range. This procedure is repeated until we find a search range that includes the real value of the available bandwidth. We then apply a similar algorithm with Pathload to search for the value of the available bandwidth.

In the case of Pathload, the search procedure stops when the width of the search range is smaller than the threshold ω. In our proposed method, we add another termination condition for the search procedure; our search procedure also stops when the measurement time exceeds τ.

Algorithms 2, 3, and 4 show the details of our method. Procedure *RuntimeLimitedPathload* is a search procedure that resembles Pathload with limited search time.

At the end of an aggregation period, the measurement result of O_iO_j at that aggregation period is calculated as follows. Assume that during that aggregation period, O_i obtained F measurement results of O_iO_j, denoted as $(A^1_L, A^1_U), (A^2_L, A^2_U), ..., (A^F_L, A^F_U)$. The measurement result of O_iO_j at that aggregation period is calculated by Eq. (5):

$$\bar{A}_{meas} = \frac{1}{F}\sum_{s=1}^{F}\frac{A^s_L + A^s_U}{2}. \tag{5}$$

4.4.3 Sending Measurement Results and Related Information

If the measurement is performed, O_i sends the result and probabilities $\Phi_{O_iO_j, O_{u_s}O_{v_s}}$ to nodes O_{u_s} $(1 \le s \le K)$. On the other hand, if the measurement is omitted, search range (S^*_L, S^*_U) is not sent to other nodes, although it is used as a measurement result of O_iO_j. This is because (S^*_L, S^*_U) is calculated based on the results of the measurements before timing t^*, that have already been sent to other nodes. Furthermore, since (S^*_L, S^*_U) is not the result of an actual measurement, it might not be near the real value of the available bandwidth of O_iO_j. Therefore, not sending it to nodes O_{u_s} $(1 \le s \le K)$ helps to avoid degradation of the measurement accuracy of $O_{u_s}O_{v_s}$.

4.4.4 Methods for Other End-to-End Measurement Tools

Our proposed method that sets parameters for each end-to-end measurement can be applied for other induced-congestion-based measurement tools such as pathChirp [15]. Similar to Pathload, pathChirp also uses several packet streams, called *chirps*, to probe the available bandwidth. However, the probing rates within a chirp increase exponentially from a lower rate L to an upper rate $U = L\gamma^{(N-1)}$, where N is the number of packets in the chirp, and γ is the *spread factor* ($\gamma > 1$). We use the same approach with the case of Pathload to set the values of L and U. We only make the following two changes for pathChirp.

First, in contrast to Pathload, the result of each pathChirp measurement is a point. Suppose that O_i stored the G measurement results of O_iO_j and its half and partial overlapping paths, denoted as $A^1, A^2, ..., A^G$ at measurement timing t^*, then values S^*_L and S^*_U in Eq. (3) are modified as follows:

$$S^*_L = \bar{A} - 1.96\sqrt{\frac{V}{G}}, \quad S^*_U = \bar{A} + 1.96\sqrt{\frac{V}{G}}. \tag{6}$$

Here, $\bar{A}$ and V are the weighted means and variances, calculated as follows:

$$\bar{A} = \sum_{s=1}^{G}\beta_s A^s, \quad V = \sum_{s=1}^{G}\beta_s A^{s2} - \bar{A}^2, \tag{7}$$

where $\beta_s = \alpha_s / \sum_{w=1}^{G}\alpha_w$ $(1 \le s \le G)$ is the weight of result A^s. When the stored data of O_i contain some measurement results of O_iO_j, we can infer that range (S^*_L, S^*_U) is near the real value of the available bandwidth and set $L = S^*_L$ and $U = S^*_U$.

Second, although we also infer that search range (S^*_L, S^*_U) is near the real value of the available bandwidth, we do not check whether its real value is inside this range, as in the case of Pathload. This is because pathChirp's algorithm does not include sending packet streams at a constant rate to check whether the real value of the available bandwidth is smaller or larger than that rate.

4.5 Implementation Issues

We discuss some issues that we must deal when implementing the proposed method.

The method for detecting path overlaps relies on the assumption that all of the routers on the paths between overlay nodes appropriately respond to `traceroute`. In the case that some of the routers do not respond to `traceroute`, we apply a similar method proposed in [24]. In particular, if path O_iO_j contains some routers between R_s and R_t that do not respond to `traceroute`, then we consider the path between R_s and R_t as a "virtual link", and apply our proposed method as usual.

Another issue is the problem of the asymmetric characteristics of the internet architecture. First, because we use the latency (RTT) to calculate the probability $\Phi_{O_{u_s}O_{v_s}, O_iO_j}$ in Eq. (2), asymmetric latency may cause error in this calculation, thus leads to the incorrect estimation of the search range. On the other hand, the asymmetric characteristics of internet routes does not effect the detection accuracy of path overlaps. This is because in our method, each overlay node uses `traceroute` to obtain only the routing information of the paths starting from itself to other overlay nodes, but not the routing information of the reverse paths.

Our method adapts to arrivals and departures of overlay nodes as follows. If an overlay node joins or leaves the network at the detection phase of path overlaps of an aggregation period, our method can immediately detect the overlapping status of the paths that start or end at the overlay node. These paths will be measured at the measurement

phase of the aggregation period as other existing paths. On the other hand, if the arrival or departure of an overlay node occurs at the measurement phase of an aggregation period, we simply ignore the measurement results of the paths that start or end at the overlay node in the current aggregation period. The new network topology will be detected in the next detection phase of path overlaps, and the paths in the new network topology will be measured as usual. In general, our method benefits from the information exchange between overlay nodes, in terms of enhancing measurement accuracy and reducing measurement traffic load. Therefore, when an overlay node joins the network, although the number of paths that need to be measured increases, because the volume of information exchange also increases, the measurement accuracy will be improved. On the other hand, the departure of an overlay node can cause degradation of measurement accuracy.

If the time taken for aggregating routing information from far nodes is larger than the length of detection phase of path overlaps, some overlapping paths can not be detected. This results in the reduction of measurement results exchanged between overlay nodes. On the other hand, if some measurement results can not be delivered at the right aggregation period, these results are not suitable for calculating the search range. Both cases results in the degradation of measurement accuracy. In a centralized method, the problem is serious, because far nodes must aggregate information to one controller. However, in our method, because the overlay node mostly exchanges information with neighboring nodes, the volume of information exchange between largely separated nodes is small, thus the problem does not cause large error.

5. Performance Evaluation

In this section, we evaluate the performance of our proposed method. We focus on evaluating how our proposed method of local information exchange relatively improves measurement accuracy rather than absolutely evaluating the measurement accuracy. Therefore, we only use evaluations based on simulations. Evaluation based on experiments in real networks is our future work. We explain the evaluation method in Sect. 5.1 and present our evaluation results and discussions in Sect. 5.2.

5.1 Evaluation Method

We compared our proposed method with an existing method [12], which we briefly explain, and make some assumptions for comparison in Sect. 5.1.1. Because measurement conflict is not avoided completely in our method, we describe a statistical model for simulating the effect of measurement conflict on the results of the end-to-end measurement in Sect. 5.1.2. We then explain the evaluation metrics and the simulation settings in Sects. 5.1.3 and 5.1.4. Throughout the simulation, we assume that Pathload is used for the end-to-end measurement of the available bandwidth. However, we expect the same trend in the evaluation results in the cases of other measurement tools.

5.1.1 Existing Method [12]

In [12], the authors proposed heuristic algorithms from graph theory to schedule different timings for the measurement tasks of overlapping paths. Although measurement conflicts are completely avoided, the measurement frequency is greatly limited, as we show below in Sect. 5.2. Two algorithms have been proposed for uniform and non-uniform measurement tasks. Uniform measurement tasks have the same running time and period, but non-uniform measurement tasks have different running times or periods. As explained in Sect. 4.3, the running time of all the measurement tasks is set to the same value τ. To keep the measurement frequency as large as possible, we set the periods of all the measurement task to the running time. That means that all the measurement tasks are uniform, and we apply the corresponding algorithm to obtain the measurement timings for each task. Because the overlay nodes do not exchange information with each other, the search range in each end-to-end measurement cannot be estimated, as in our proposed method. We therefore set the search range for path O_iO_j to $(0, C_{O_iO_j})$, where $C_{O_iO_j}$ is the capacity of the first IP link of path O_iO_j.

5.1.2 Simulating the Effect of Measurement Conflict on the End-to-End Measurement Result

A measurement conflict may occur when the measurements of overlapping paths are performed concurrently. When it occurs, we cannot determine exactly the effect it causes on the measurement results. However, we can statistically simulate this effect based on the mechanism of end-to-end measurement as follows.

As mentioned above, we adopt a mechanism similar to Pathload for measuring the end-to-end available bandwidth. In particular, at each iteration of the measurement, the source node sends a string of packet streams (a packet fleet) with a predetermined rate and checks whether there is an increasing trend in the one-way delays to judge if the real value of the available bandwidth is larger or smaller than the rate. This judgement may become incorrect if the durations of sending packet streams overlap with those of overlapping paths.

We assume that at measurement timing t^* of path O_iO_j, H ($H \geq 1$) overlapping paths of O_iO_j also perform measurements. For simplicity, we denote path O_iO_j as path 1 and each of these H overlapping paths as path p ($2 \leq p \leq H+1$). For path p ($1 \leq p \leq H+1$), the duration for sending one packet stream is denoted as θ_p, and the interval between two successive packet streams in each packet fleet is set to ψ_p. $\pi_p = \theta_p + \psi_p$ is the period of one packet stream of path p. Set $\phi_p = \theta_p/\pi_p = \theta_p/(\theta_p + \psi_p)$, $0 < \phi_p < 1$. In general, θ_p is much smaller than ψ_p [4], and thus $\phi_p \ll 1$.

We calculate the probability that the duration of send-

ing packet streams of path 1 overlaps with the durations of sending packet streams of any path p ($2 \leq p \leq H+1$) as follows. During one period π_1 of one packet stream of path 1, the duration of sending packet streams of path p ($2 \leq p \leq H+1$) becomes $t_p = \phi_p \pi_1$. The probability that the durations of sending packet streams of paths 1 and p overlap with each other during one period π_1 is $\gamma = \theta_1/(\pi_1 - t_p) = \phi_1/(1-\phi_p)$. Since $\phi_p \ll 1$, we have $\gamma \approx \phi_1$. Therefore, the probability that the duration of sending packet streams of path 1 overlaps with the durations of sending packet streams of any path p ($2 \leq p \leq H+1$) becomes $1-(1-\phi_1)^H$.

We then simulate the effect of the measurement conflict on the result of the end-to-end measurement of path O_iO_j at measurement timing t^* as follows. We randomly decide that the durations of sending packet streams of path O_iO_j overlaps with the durations of sending packet streams of overlapping paths of O_iO_j with probability $1-(1-\phi_1)^H$. In this case, we randomly decide whether the sending rate is larger or smaller than the real value of the available bandwidth.

For simplicity, we use the default settings of Pathload [4] to set $\phi_p = 0.1$ for all paths.

5.1.3 Evaluation Metrics

We compare our proposed method and the method in [12] with the following metrics:

- Metrics related to measurement accuracy
  We use the relative error of the measurement results as a metric to evaluate the measurement accuracy of the methods. As explained in Sect. 2, the measurement accuracy is largely decided by the measurement frequency and conflict. Therefore, we evaluate the following metrics:
  - *Relative error*
    The relative error is calculated by Eq. (1), where $\bar{A}_{meas}$ is the average of the measurement results in an aggregation period, defined by Eq. (5), and $\bar{A}$ is the average of the real value of the available bandwidth in that aggregation period.
  - *Measurement frequency* and *number of measurements*
    The measurement frequency of a path at an aggregation period is calculated by:
    $$f = \frac{h\tau}{\Delta}, \tag{8}$$
    where h is the number of measurements in that aggregation period, τ is the duration of one measurement, and Δ is the duration of that aggregation period.
  - *Measurement conflict rate*
    The measurement conflict rate is defined as the ratio of the number of measurements that experience conflicts to the number of measurements, at one aggregation period.

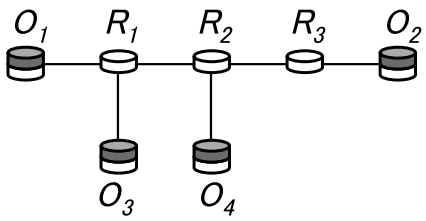


Fig. 4 Small network topology.

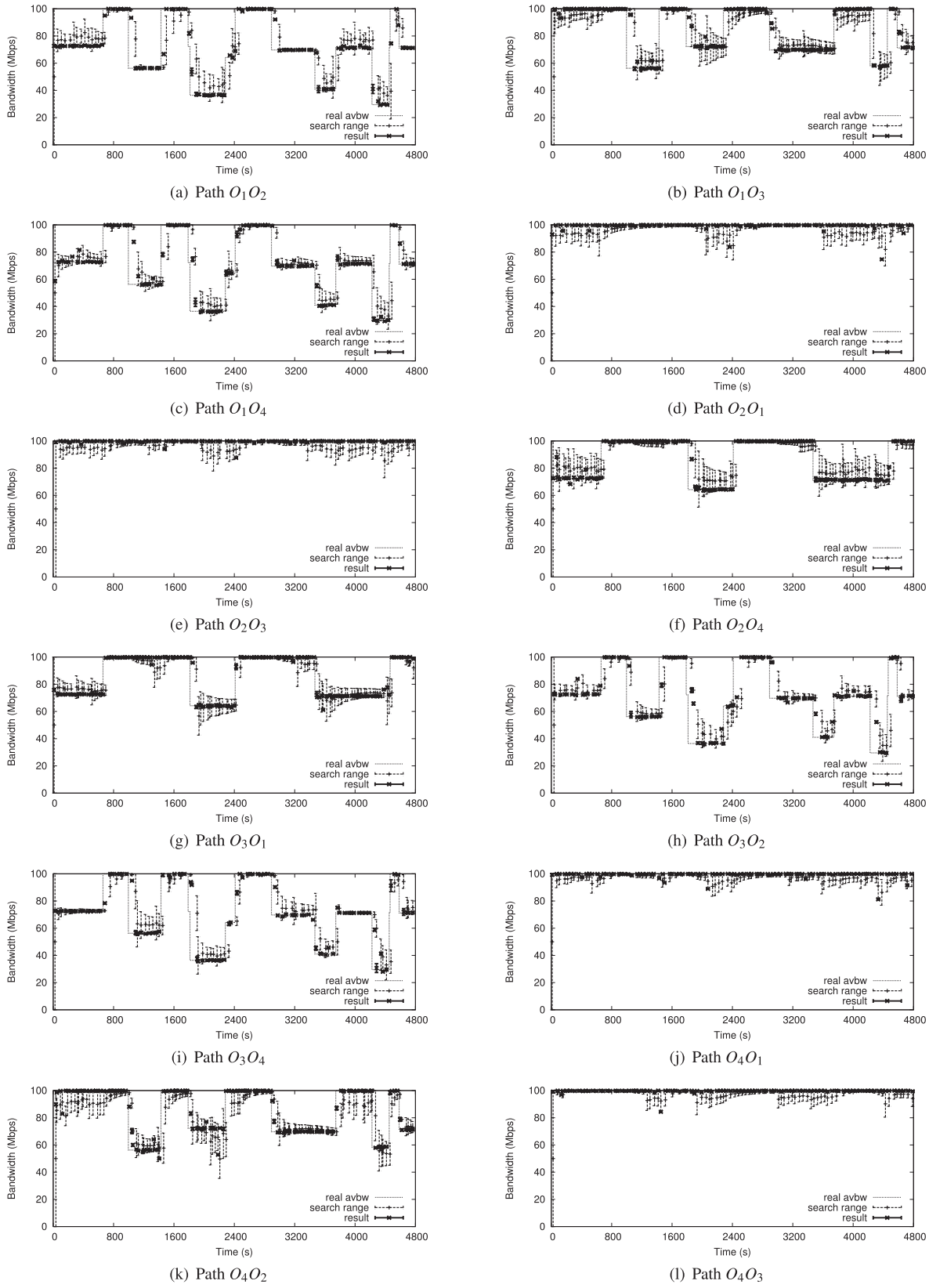
- Metrics related to measurement traffic load
  We use the average number of packet fleets traversing one link to evaluate the measurement traffic load.

5.1.4 Simulation Settings

The purposes of our simulations are to confirm that the proposed method works properly as designed and to compare its performance and that of the method in [12]. For the former purpose, we applied our method to the small network topology shown in Fig. 4 and observed its detailed behavior. For the latter purpose, we used three types of large network topologies: the AT&T topology [25], generated topologies based on the Barabasi-Albert (BA) [26], and the Waxman models [27]. We generated ten topologies for each model using the BRITE topology generator [28]. All topologies have 523 nodes and 1304 links. We set the density of the overlay nodes to 0.2 and randomly chose them from 523 nodes. For averaging the results, the choice of the overlay nodes was taken 100 times for the AT&T topology and ten times for each topology of the BA and Waxman models. For simplicity, we assume that the capacity of all IP links in the network is C and set $C = 100$ [Mbps].

We made the following assumptions about the temporal changes in the traffic amount between the overlay nodes. Assume that cross traffic occurs at fraction α ($0 < \alpha \leq 1$) of the paths. For the small network topology, α was set to 0.2, and in the large network topologies, it was set to 0.02. For path O_iO_j where cross traffic occurs, denote its IP links as $l_1, l_2, ..., l_r$. Assume that among the paths where cross traffic occurs, the number of paths that share the link l_t ($1 \leq t \leq r$) is b_t. Set $b_{max} = \max\{b_1, b_2, ..., b_r\}$. Furthermore, set $s_{max} = 0.9C/b_{max}$, $s_{min} = 0.5s_{max}$. The rate of cross traffic of O_iO_j was then randomly chosen in range $[s_{min}, s_{max}]$. Furthermore, the intervals where traffic occurs and does not occur were randomly chosen in range [120 s, 1200 s].

For the parameters related to end-to-end measurement, because we adopt a method similar to Pathload, we set the parameters based on the suggestions of the authors of Pathload [4]. In particular, we set $\tau = 12$ [s] and $\omega = 400$ [kbps]. The parameter for omitting measurement ϵ was set to 2ω. Minimum value D of the measurement times of one path at an aggregation period was set to 2. For the small network topology, we set the measurement duration to 400τ. On the other hand, in the large network topologies, the measurement duration contained ten aggregation periods, and each aggregation period was set to 1200τ.



(a) Path O_1O_2 (b) Path O_1O_3

(c) Path O_1O_4 (d) Path O_2O_1

(e) Path O_2O_3 (f) Path O_2O_4

(g) Path O_3O_1 (h) Path O_3O_2

(i) Path O_3O_4 (j) Path O_4O_1

(k) Path O_4O_2 (l) Path O_4O_3

Fig. 5 Measurement result in small network topologies.

Table 1 Distribution of relative errors.

Topology / Relative error / Method	AT&T				BA				Waxman			
	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	56.600%	32.184%	9.576%	1.432%	50.994%	29.480%	9.542%	1.153%	33.411%	15.373%	3.418%	0.167%
Proposed method	41.999%	18.087%	3.260%	0.194%	35.472%	14.161%	2.546%	0.105%	26.841%	9.492%	1.385%	0.024%

Our simulation program was written in C language and run on commodity Linux machines with default settings.

5.2 Evaluation Results and Discussions

5.2.1 Evaluation Results in Small Network Topologies

Figure 5 shows the measurement results of all paths in the network in Fig. 4. The dotted lines show the real values of the available bandwidth, the dashed bars show the search ranges, and the thick bars show the measurement results. Because our measurement accuracy depends on the search range, we consider the variation of the search range to evaluate the effectiveness of our method. As shown in Fig. 5, the search range varies based on the real value of the available bandwidth and tends to approach it. When its real value changes greatly, the width of the search range is large at first, but it gradually becomes small, and the search range quickly approaches the real value of the available bandwidth. These results show that our proposed method for calculating search range is efficient for measuring available bandwidth.

5.2.2 Evaluation Results in Large Network Topologies

Measurement accuracy

Table 1 shows the distribution of the relative errors in the measurement results in the AT&T, BA, and Waxman topologies. In particular, it shows the percentage of relative errors in the measurement results that are not smaller than 0.05, 0.1, 0.2, and 0.4. Table 2 shows the average values of the relative errors in the measurement results. The relative errors in the measurement results of our method are approximately only 65% of those in the method in [12]. To explain these results, we use the evaluation results of the average number of measurements, the average measurement frequencies and the average measurement conflict rates of an overlay path during an aggregation period, shown in Tables 3, 4 and 5, respectively. The number of measurements and measurement frequency in our method are much larger than those of the method in [12], but only about 12% of the measurements experience conflicts. Therefore, our method's measurement accuracy surpasses the method in [12]. We also observe in Tables 1 and 2 that the Waxman topology has smaller relative error than the AT&T and BA topologies for the following reason. From the simulation results, we found fewer half and partial overlapping paths in the Waxman topology than in the AT&T and BA topologies. Therefore, the measurement frequency is the largest, meaning that the number of measurements is the largest, and thus the relative error is the smallest in the Waxman topology.

Table 2 Average relative errors.

Method / Topology	AT&T	BA	Waxman
Existing method	0.088	0.081	0.049
Proposed method	0.058	0.049	0.039

Table 3 Average number of measurements during an aggregation period.

Method / Topology	AT&T	BA	Waxman
Existing method	3.287	5.912	13.202
Proposed method	11.050	20.259	28.388

Table 4 Average measurement frequencies during an aggregation period.

Method / Topology	AT&T	BA	Waxman
Existing method	2.739×10^{-3}	4.927×10^{-3}	11.002×10^{-3}
Proposed method	9.208×10^{-3}	16.883×10^{-3}	23.657×10^{-3}

Table 5 Average measurement conflict rates during an aggregation period.

Method / Topology	AT&T	BA	Waxman
Existing method	0.000	0.000	0.000
Proposed method	0.141	0.107	0.119

Table 6 Average number of packet fleets traversing one link during an aggregation period.

Method / Topology	AT&T	BA	Waxman
Existing method	634.483	510.533	828.918
Proposed method	1508.375	1374.447	1411.387

Table 7 Average number of packet fleets traversing one link at one measurement.

Method / Topology	AT&T	BA	Waxman
Existing method	6.000	6.000	6.000
Proposed method	5.814	5.784	5.674

Measurement traffic load

Tables 6 and 7 show the average number of packet fleets traversing one link during an aggregation period and the average number of packet fleets traversing one link at one measurement. As shown in Table 6, in our method, the average number of packet fleets traversing one link during an aggregation period is larger than that in the method in [12]. This is

Table 8 Distribution of relative errors when measurement traffic loads of existing and proposed methods are identical.

Topology	AT&T				BA				Waxman			
Relative error / Method	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	56.524%	32.274%	9.7203%	1.404%	51.792%	30.487%	10.095%	1.244%	34.835%	16.453%	3.848%	0.188%
Proposed method	54.095%	28.301%	6.960%	0.621%	50.120%	26.604%	6.923%	0.534%	34.283%	13.938%	2.279%	0.048%

because the measurement frequency in our method is much larger (Table 4). However, the average number of packet fleets traversing one link at one measurement in our method is smaller than that of the method in [12] for the following reason. In the method in [12], because the search range in each end-to-end measurement of the available bandwidth is set to $(0, C)$, the number of packet fleets at one measurement is constant for all measurements. On the other hand, in our proposed method, since the search range is calculated based on the measurement results exchanged between overlay nodes, it is narrower and nearer the real value of the available bandwidth than the search range in the method in [12]. Therefore, the number of packet fleets at one measurement is smaller, meaning that the traffic load of each measurement is smaller in our proposed method.

Evaluation results with equal measurement traffic load

Next, we adjust the measurement frequencies in our proposed method so that its measurement traffic loads and those of the method in [12] are the same. Tables 8 and 9 show the evaluation results of the relative errors in the measurement results in the AT&T, BA, and Waxman topologies. Table 10 shows the average number of packet fleets traversing one link during an aggregation period of the method in [12] and our proposed method, which are almost the same, as expected. Table 11 shows the average number of packet fleets traversing one link at one measurement of the method in [12] and the proposed method. Even though the number of packet fleets traversing one link at one measurement in [12] is unchanged, that value in our proposed method is slightly larger than before adjusting the measurement frequencies. This is because the measurement frequency in our method is reduced, thus the number of measurement results used for calculating parameters of each measurement also decreases, and the resulting search ranges are not near the real value of available bandwidths as before adjusting the measurement frequencies.

However, as shown in Tables 8 and 9, the measurement accuracy of the proposed method is slightly better than that of the method in [12]. This is because the number of measurements and the measurement frequency in our method remain larger than those of the method in [12], as shown in Tables 12 and 13, the measurement conflict rate is small, as shown in Table 14, and the search range in our proposed method is narrower and nearer the real value of the available bandwidth than the method in [12], as explained above.

Table 9 Average relative errors when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	0.088	0.083	0.051
Proposed method	0.077	0.073	0.046

Table 10 Average number of packet fleets traversing one link during an aggregation period when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	655.325	527.841	822.997
Proposed method	652.432	526.810	821.838

Table 11 Average number of packet fleets traversing one link at one measurement when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	6.000	6.000	6.000
Proposed method	5.875	5.867	5.730

Table 12 Average number of measurements during an aggregation period when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	3.562	5.777	12.713
Proposed method	3.736	5.966	13.345

Table 13 Average measurement frequencies during an aggregation period when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	2.968×10^{-3}	4.814×10^{-3}	10.594×10^{-3}
Proposed method	3.113×10^{-3}	4.972×10^{-3}	11.121×10^{-3}

Table 14 Average measurement conflict rates during an aggregation period when measurement traffic loads of existing and proposed methods are identical.

Method / Topology	AT&T	BA	Waxman
Existing method	0.000	0.000	0.000
Proposed method	0.124	0.095	0.114

6. Conclusion

In this paper, we proposed a distributed measurement

method for available bandwidth in overlay networks that reduces measurement conflicts by detecting path overlaps and then appropriately adjusting the measurement frequencies and the measurement timings of overlay paths. We also proposed a method to improve measurement accuracy while reducing the traffic load of each measurement by exchanging the measurement results among neighboring overlay nodes. Simulation results show that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. The measurement accuracy of our method remains better than the existing method when the total measurement traffic loads of both methods are equal.

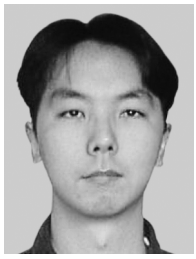
Acknowledgment

This work was supported in part by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

References

[1] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," IEEE J. Sel. Areas Commun., vol.25, no.9, pp.1706–1716, 2007.
[2] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," SIGMETRICS Perform. Eval. Rev., vol.36, no.1, pp.313–324, June 2008.
[3] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," Proc. ACM SIGCOMM 2002, Aug. 2002.
[4] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," Proc. PAM 2002, pp.14–25, 2002.
[5] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," Proc. ACM SIGCOMM 2003, 2003.
[6] C. Tang and P. McKinley, "On the cost-quality tradeoff in topology-aware overlay path probing," Proc. ICNP 2003, Nov. 2003.
[7] N. Hu and P. Steenkiste, "Exploiting internet route sharing for large scale available bandwidth estimation," Proc. IMC 2005, Oct. 2005.
[8] C. Xing, M. Chen, and L. Yang, "Predicting available bandwidth of internet path with ultra metric space-based approaches," Proc. IEEE GLOBECOM 2009, pp.1–6, 2009.
[9] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella, "On the treeness of internet latency and bandwidth," Proc. ACM SIGMETRICS 2009, pp.61–72, 2009.
[10] S. Song, P. Keleher, B. Bhattacharjee, and A. Sussman, "Decentralized, accurate, and low-cost network bandwidth prediction," Proc. IEEE INFOCOM 2011, pp.6–10, 2011.
[11] P. Calyam, C. Lee, E. Ekici, M. Haffner, and N. Howes, "Orchestration of network-wide active measurements for supporting distributed computing applications," IEEE Trans. Comput., vol.56, no.12, pp.1629–1642, Dec. 2007.
[12] M. Fraiwan and G. Manimaran, "Scheduling algorithms for conducting conflict-free measurements in overlay networks," Comput. Netw., vol.52, pp.2819–2830, 2008.
[13] Z. Qin, R. Rojas-Cessa, and N. Ansari, "Task-execution scheduling schemes for network measurement and monitoring," Comput. Commun., vol.33, no.2, pp.124–135, 2010.
[14] T.H. Dinh, G. Hasegawa, and M. Murata, "A low-cost, distributed and conflict-aware measurement method for overlay network services utilizing local information exchange," IEICE Trans. Commun., vol.E96-B, no.2, pp.459–469, Feb. 2013.
[15] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," Proc. PAM 2003, 2003.
[16] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," IEEE J. Sel. Areas Commun., vol.21, no.6, pp.879–894, Aug. 2003.
[17] W. Tan, M. Zhanikeev, and Y. Tanaka, "Rate-based and gap-based available bandwidth estimation techniques in cross-traffic context," Proc. APNOMS 2006, pp.73–81, 2006.
[18] C.D. Guerrero and M.A. Labrador, "On the applicability of available bandwidth estimation techniques and tools," Comput. Commun., vol.33, no.1, pp.11–22, 2010.
[19] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," Proc. IEEE INFOCOM 2002, pp.170–179, June 2002.
[20] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," Proc. ACM SIGCOMM 2004, pp.15–26, Aug. 2004.
[21] N.M.M.K. Chowdhury and R. Boutaba, "A survey of network virtualization," Comput. Netw., vol.54, no.5, pp.862–876, 2010.
[22] J. Rubio-Loyola, A. Galis, A. Astorga, J. Serrat, L. Lefevre, A. Fischer, A. Paler, and H. Meer, "Scalable service deployment on software-defined networks," IEEE Commun. Mag., vol.49, no.12, pp.84 –93, Dec. 2011.
[23] C.L.T. Man, G. Hasegawa, and M. Murata, "Monitoring overlay path bandwidth using an inline measurement technique," IARIA International Journal on Advances in Systems and Measurements, vol.1, no.1, pp.50–60, 2008.
[24] Y. Chen, D. Bindel, H. Song, and R. Katz, "An algebraic approach to practical and scalable overlay network monitoring," Proc. ACM SIGCOMM 2004, Aug. 2004.
[25] N. Spring, R. Mahajan, and C. Wetherall, "Measuring isp topologies with rocketfuel," Proc. ACM SIGCOMM 2002, Jan. 2002.
[26] A. Barabasi and R. Albert, "Emergence of scaling in random networks," Science, vol.286, pp.509–512, Oct. 1999.
[27] B.M. Waxman, "Routing of multipoint connections," IEEE J. Sel. Areas Commun., vol.6, pp.1617–1622, Dec. 1988.
[28] BRITE: Boston university Representative Internet Topology gEnerator, available at http://www.cs.bu.edu/brite/index.html

Tien Hoang Dinh received the M.E. degree in Information Science and Technology from Kyoto University, Japan, in 2007. He is now a D.E. candidate at Graduate School of Information Science and Technology, Osaka University. His research interest includes overlay networks and network measurement.



Go Hasegawa received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1997 and 2000, respectively. From July 1997 to June 2000, he was a Research Assistant of Graduate School of Economics, Osaka University. He is now an Associate Professor of Cybermedia Center, Osaka University. His research work is in the area of transport architecture for future high-speed networks and overlay networks. He is a member of the IEEE.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Science from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor of Cybermedia Center, Osaka University, and is now with Graduate School of Information Science and Technology, Osaka University since April 2004. He has more than five hundred papers of international and domestic journals and conferences. His research interests include computer communication network architecture, performance modeling and evaluation. He is a member of IEEE, ACM and IEICE. He is a chair of IEEE COMSOC Japan Chapter since 2009. Also, he is now partly working at NICT (National Institute of Information and Communications Technology) as Deputy of New-Generation Network R&D Strategic Headquarters.