2017 IEEE International Parallel and Distributed Processing Symposium Workshops

Reliability Calculation of P2P Streaming Systems with Bottleneck Links

Satoshi Fujita
Department of Information Engineering, Hiroshima University

***Abstract*—Let G be a network with node set V and link set E, where each link e in E is associated with capacity $c(e)$ and failure probability $p(e)$. Let $D = (s, t, d)$ be a flow demand on G which requests that a video stream of bit rate d should be delivered from source node s to sink node t through network G. The reliability of network G with respect to flow demand D is the probability of surviving a subgraph of G which admits D under the probabilistic link failures determined by function p. This paper shows that if G contains a constant number of *bottleneck links* and if the given video stream is divided into constant number of sub-streams, we can calculate the reliability of G in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time, where $\alpha|E|$ is the number of links in the maximum connected component obtained by removing bottleneck links. This is a significant improvement of a naive method which takes $\mathcal{O}(2^{|E|}|V||E|)$ time.**

***Index Terms*—Flow network, reliability, NP-hardness, exponential time algorithm.**

I. Introduction

Let G be a Peer-to-Peer (P2P) video streaming system consisting of node set V and link set E. Each link $e \in E$ is associated with capacity $c(e) \in \mathbb{N}$ and failure probability $p(e) \in [0, 1)$, which means that *link e can carry a stream of bit rate $c(e)$ while it is out of use with probability $p(e)$*. No relevance is assumed between functions c and p. Let $D = (s, t, d)$ be a **flow demand** on G which requests that a video stream of bit rate d should be delivered from source node s to sink node t through network G, where the given stream can be divided into d sub-streams of unit bit-rate which can reach t through different delivery paths.

In this paper, we consider the problem of calculating the probability of surviving a subgraph of G which admits a given flow demand D under the constraint on functions c and p. Such a surviving probability is called the **reliability** of G with respect to flow demand D. This problem is known to be $\mathcal{NP}$-hard [8] which implies that the calculation of the reliability takes exponential time unless $\mathcal{P} = \mathcal{NP}$. The simplest way to calculate the reliability is to examine all of the $2^{|E|}$ configurations of link failures and to sum up the probability of the occurrence of configurations which admit D. More concretely, as is illustrated in Figure 1, for each configuration of available links $E' \subset E$, we identify whether or not the subgraph of G induced by E' admits D by applying a maximum flow algorithm, and if it admits the flow, we increase the reliability by

$$\prod_{e \in E'} p(e) \times \prod_{e \in E - E'} (1 - p(e)).$$

Clearly, such a naive algorithm takes long computation time proportional to $2^{|E|}$ since we need to consider all subsets E' of E in the worst case. Our main contribution in the current paper is to reduce the exponent $|E|$ in the above expression by considering "bottleneck" existing in the given graph G. The basic idea is to consider a set of (bottleneck) links so that the removal of those links disconnects s and t and divides G into two connected components consisting of at most $\alpha|E|$ links for some $0 < \alpha < 1$. We then prove that the time required for the reliability calculation for the whole graph reduces to $\mathcal{O}(2^{\alpha|E|}|V||E|)$ if the number of bottleneck links and the amount of flow demand d are both constant.

The remainder of this paper is organized as follows. Section II overviews related works. Sections III and IV describes the details of the proposed algorithm. Finally, Section V concludes the paper with future work.

II. Related Work

P2P video streaming systems are widely studied in recent years [2], [18], [9], [11], [12], [19]. Those systems can be classified into several types by the way of delivering video streams to the subscribers. In mesh-based systems such as Bullet [10], PRIME [13] and CoolStreaming/DONet [17], each peer establishes neighborhood relationships to random peers which are selected from the set of peers depending on the upload capacity and the content availability, and periodically exchanges the content availability with its neighborhood to "pull" missing content from the neighbors. Such a dynamic

This work was supported by KAKENHI, the Grant-in-Aid for Scientific Research (B), Grant Number 16H02807.

Kagamiyama 1-4-1, Higashihiroshima, Hiroshima, 739-8527 Japan, fujita@se.hiroshima-u.ac.jp

978-1-5386-3408-0/17 $31.00 © 2017 IEEE
DOI 10.1109/IPDPSW.2017.20

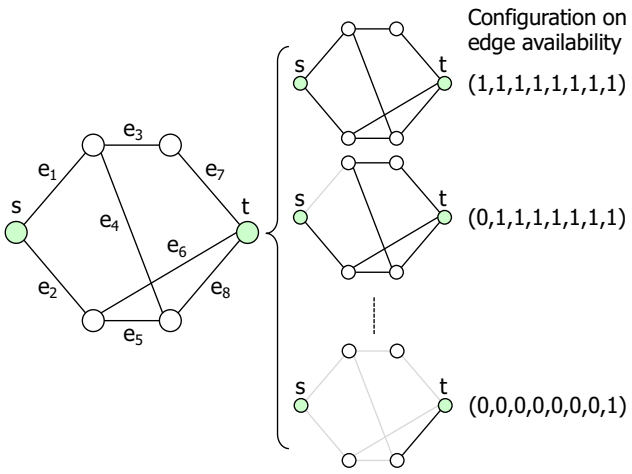

Fig. 1. Naive calculation of the reliability of a graph.

construction of random overlay is robust against peer churns, but it is difficult to guarantee the quality of content distribution such as the delay, jitter, and the transmission overhead. In addition, in mesh-based systems, a high capacity peer is likely being selected as a neighbor of many peers which would cause a bottleneck at such peers.

In tree-based systems [5], [15], [4], [20], on the other hand, peers are organized in a tree-structured overlay. In those systems, a video stream is "pushed" by the media server located at the root of the tree and is delivered to the downstream peers by repeating store-and-relay operation. Tree-based systems are simple and efficient but are not robust against peer churn. In addition, it could not fully utilize contributed resources since it does not use the upload bandwidth of leaf peers in the overlay. Such drawbacks of tree-based systems can be overcome by adopting *multiple trees* [14], [3], [16] instead of single tree. More concretely, by dividing a given video stream into multiple sub-streams and by delivering those sub-streams using different spanning trees, we could significantly improve the fault-tolerance and the resource utilization, provided that we can arrange spanning trees in such a way that most of the peers are internal in only one tree and leaf peers in others [1], [3], [6].

The reliability of P2P video streaming systems is generally evaluated by the availability and the stability of delivery paths in the P2P overlay which could be obtained by calculating the probability of surviving a given path in an appropriate probability space. However, as is described above, the delivery paths of video streams are *not* fixed in mesh-based systems and the same is true for several tree-based systems which have an ability of adjusting the route to subscribers after detecting faults in

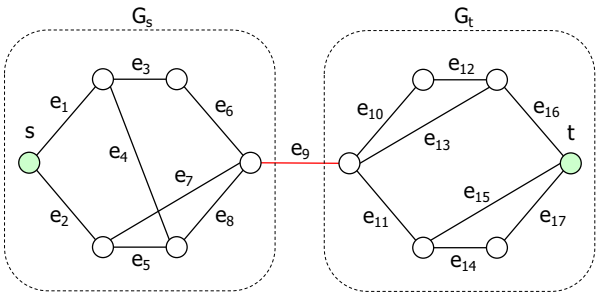

Fig. 2. A graph with bridge (red link e_9 is bridge connecting G_s and G_t).

the underlying P2P overlay. In addition, such a routing of video streams is not conducted through a simple path in multiple-tree-structured overlays, which makes the analysis of the availability more complicated and difficult. An idea to overcome such a difficulty is to model the system to push a constant number of sub-flows from the source to the sink and to analyze the reliability of the network concerned with sub-flows. Although the calculation of the reliability of network concerned with general flow is know to be $\mathcal{NP}$-hard [8], it is meaningful to reduce the exponent as is widely done in many previous works concerned with exponential algorithms [7].

III. Proposed Algorithm

A. Overview

In this paper, we consider a class of graphs to have small number of bottleneck links. Link set $E^* \subset E$ is called a set of α-**bottleneck links** of G with respect to s and t if:

1) the removal of E^* from G disconnects s and t while the removal of any proper subset of E^* does not disconnect s and t (i.e., it is a minimal subset to disconnect s and t);
2) the cardinality of E^* is a constant; and
3) each connected component obtained by removing E^* from G contains at most $\alpha|E|$ links.

Graph with a bridge link e^* is a typical example of such graphs. Note that such a restriction on the class of graphs reduces the computation time of combinatorial problems in general. In fact, if the removal of e^* disconnects s and t, and two connected components G_s and G_t obtained by the removal consist of at most $\alpha|E|$ links, where G_s is the side of containing the source s, we can calculate the reliability of G in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time as follows (see Figure 2 for illustration):

- If $c(e^*) < d$, the reliability of G with respect to the demand is trivially zero. Thus in the following, we assume $c(e^*) \geq d$, without loss of generality.
- Let x, y be end-vertices of link e^*, where x and y are vertices in G_s and G_t, respectively.
- Calculate the reliability $r(G_s)$ of G_s with respect to flow demand (s, x, d) and calculate the reliability $r(G_t)$ of G_t with respect to flow demand (y, t, d).
- Then, the reliability r of G with respect to D from s to t is calculated as

$$r \;=\; r(G_s) \times p(e^*) \times r(G_t). \qquad (1)$$

Since a naive algorithm calculates $r(G_s)$ and $r(G_t)$ in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time, this algorithm calculates the $r(G)$ in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time.

More generally, as will be described below, for any graph G with a set of α-bottleneck links, we can calculate the reliability of G in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time as long as the given stream is divided into constant number of sub-streams. In the following, we describe the detail of the proposed algorithm. The explanation consists of three parts. In the first part, we define a set of subproblems defined by *different assignments of the flow demand to bottleneck links* (Section III-B). In the second part, after providing a description of the data structure, we describe how to update the data structure while solving a series of subproblems defined in the first part (Section III-C). The third part is the core of the proposed algorithm (Section IV). In the third part, we describe a way of accumulating the calculation results for subproblems using two techniques; i.e., a classification of subproblems with the notion of supporting assignment and an accumulation using inclusion-exclusion principle. The reader should remind that due to the minimality of subset E^* of α-bottleneck links, the number of connected components obtained by removing E^* from G is exactly two.

B. Subproblems to be Solved

Let $E^* = \{e_1, \ldots, e_k\}$ be a set of α-bottleneck links of G. When $k = 1$, we may simply check whether or not the unique link in E^* has enough capacity to admit the given flow demand, and if it has enough capacity, we may take a product of probabilities of three independent events as in Equation (1).

When $k \geq 2$, however, we should consider different assignments of sub-streams into k links and examine the reliability for each of those assignments; in other words, each assignment defines a subproblem to be solved. Note that although those subproblems are not independent in general, the number of different assignments is bounded by a constant as long as both d and k are constant. In fact, in any assignment of d sub-streams of unit bit-rate, each link $e \in E^*$ is assigned at most $\min\{c(e), d\}$ sub-streams. Hence the number of different assignments is at most d^k, which is constant provided that both d and k are constant.

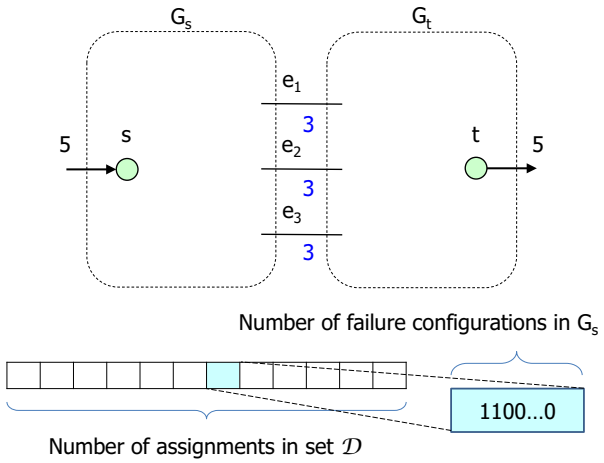

Fig. 3. Example 1 and example 2.

In the following, we denote the set of all such assignments by $\mathcal{D}$ and represent each assignment in $\mathcal{D}$ by a k-tuple $(a_1, a_2, \ldots, a_k)$ indicating that d sub-streams derived from given stream is distributed over k links in E^* so that link e_i is assigned a_i sub-streams.

Example 1: When $d = 5$ and $E^* = \{e_1, e_2, e_3\}$ with $c(e_1) = c(e_2) = c(e_3) = 3$, the set of assignments $\mathcal{D}$ is given as follows

$$\begin{aligned}\mathcal{D} \;=\; \{&(0,2,3),(0,3,2),\\ &(1,1,3),(1,2,2),(1,3,1),\\ &(2,0,3),(2,1,2),(2,2,1),(2,3,0),\\ &(3,0,2),(3,1,1),(3,2,0)\}.\end{aligned}$$

C. Data Structure

Let $G_s = (V_s, E_s)$ and $G_t = (V_t, E_t)$ be two connected components obtained by removing E^* from G. In the following, we will merely describe the details of the data structure for G_s since the data structure for G_t can be realized in a similar manner. The proposed data structure is an array of length $2^{|E_s|}$. Elements in the array correspond to failure configurations in component G_s numbered from 0 to $2^{|E_s|} - 1$, and each element in the array records "for which assignment in $\mathcal{D}$, the corresponding configuration successfully delivers the stream of bit-rate d to subgraph G_t through E^*." Such information is represented as a $|\mathcal{D}|$-bit sequence indicating a subset of $\mathcal{D}$; e.g., $000\ldots0$ indicates the empty set and $111\ldots1$ indicates set $\mathcal{D}$.

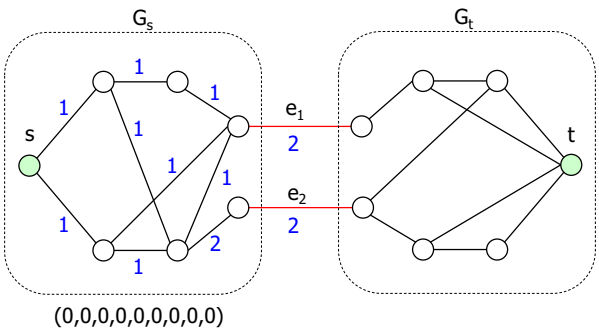

Fig. 4. A graph with two bottleneck links e_1 and e_2 (the number attached to each link represents the link capacity and the vector $(0,0,0,0,0,0,0,0,0,0)$ indicates that it corresponds the configuration with no link failure).

Example 2: Consider the set of assignments $\mathcal{D}$ derived in Example 1. Since $|\mathcal{D}| = 12$, each element in the array is a binary sequence of length 12. If the i^{th} element in the array has value 110000000000, then this indicates that the i^{th} failure configuration in G_s admits the delivery of given stream from s to G_t under the first and the second assignments in $\mathcal{D}$; i.e., assignments $(0,2,3)$ and $(0,3,2)$. See Figure 3 again for illustration.

Such an array of binary sequences can be obtained by solving the max-flow problem for each pair of an assignment in $\mathcal{D}$ and a failure configuration in G_s. More concretely, the outcome of a maximum flow problem is written to the array in such a way that: 1) if the examined problem is for the i^{th} failure configuration, then the result is written to the i^{th} element in the array; 2) if the examined problem is for the j^{th} assignment in set $\mathcal{D}$, then the result is written to the j^{th} bit in the element; and 3) if the result of examination is "Yes" then write value "1" to the corresponding bit and if the result of examination is "No" then write value "0" to the corresponding bit. The number of invocations of a max-flow algorithm is $\mathcal{D}\times 2^{|E_s|}$ for subgraph G_s and $\mathcal{D}\times 2^{|E_t|}$ for subgraph G_t. Thus we can calculate the array for G_s in $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time by using a max-flow algorithm of $\mathcal{O}(|V||E|)$ time.

IV. Efficient Accumulation of Probabilities

The proposed algorithm calculates the reliability of G by *accumulating* the above two arrays. Before describing the details of the accumulation process, we explain why a simple multiplication of the probabilities as in Equation (1) does not work in general.

Example 3: Figure 4 shows a graph separated by two bottleneck links e_1 and e_2. The number attached to each link represents the link capacity, and we will assume that

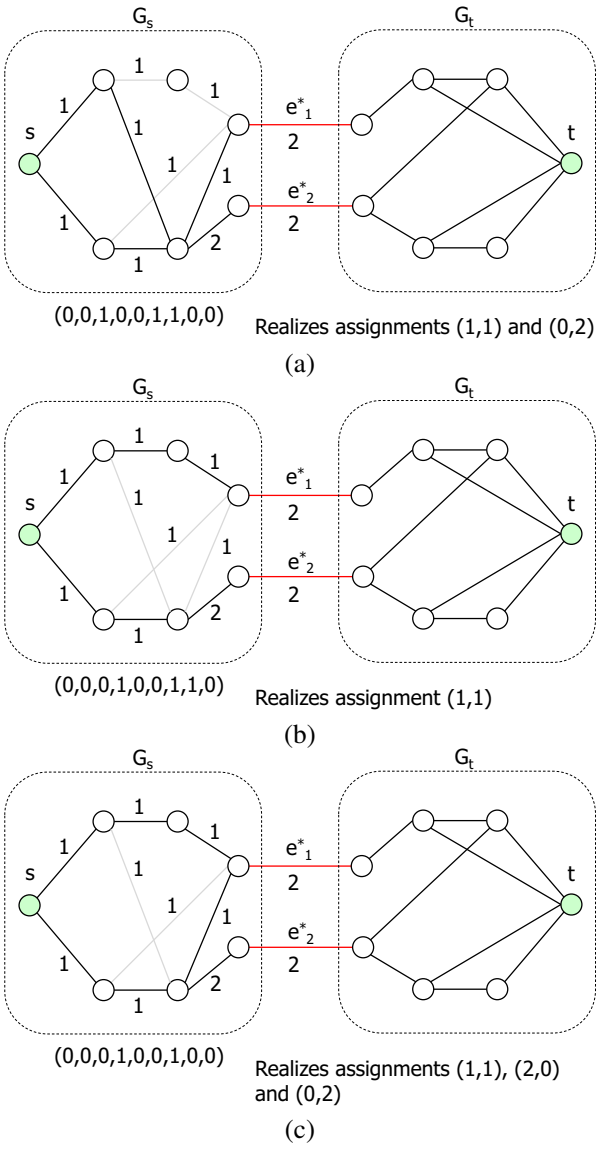

Fig. 5. Three configurations of link failures in subgraph G_s.

the graph admits a flow demand of amount two from s to t when all links are available. As for the usage of bottleneck links to *realize* a flow of amount two from G_s to G_t, we can consider three assignments of the flow on the bottleneck links, i.e.,

$$\mathcal{D} = \{(2,0),(1,1),(0,2)\}.$$

Such a usage of bottleneck links is *independent* of the probability of link failures. Figure 5 shows three typical failure configurations, which *realize* different sets of assignments; i.e., the first configuration realizes two assignments $(1,1)$ and $(0,2)$, the second configuration realizes one assignment $(1,1)$, and the third configuration realizes three assignments $(1,1)$, $(2,0)$ and $(0,2)$. Since

the set of assignments realized by those configurations intersect with each other in a complicated manner, we can not directly multiply the probability of link failures in a configuration in G_s with the probability of link failures in a configuration in G_t, as in Equation (1).

To overcome such a difficulty, we introduce the notion of "supporting subsets" of bottleneck links.

Definition 1: A subset E' of bottleneck links E^* is said to **support** assignment $(a_1, a_2, \ldots, a_k) \in \mathcal{D}$ if $a_i > 0$ implies $e_i \in E'$.

Example 4: Let $k = 3$. Then a subset $\{e_1, e_3\}$ of bottleneck links supports assignments $(2, 0, 1)$ and $(3, 0, 4)$ but does not support assignment $(1, 1, 0)$.

By definition, the set of bottleneck links E^* supports all assignments in set $\mathcal{D}$ and the empty set $\emptyset$ supports no assignment in $\mathcal{D}$. Note that the configuration of link failures of bottleneck links designated by subset E' happens with probability

$$p_{E'} = \prod_{e \in E'} p(e) \times \prod_{e \in E^* - E'} (1 - p(e)) \qquad (2)$$

which will be used at the last step of the calculation of the reliability of G as in the case of graphs with a bridge.

A. Classification of Assignments

At first, we classify assignments in $\mathcal{D}$ into several subsets so that each subset consists of assignments supported by the subset E' of E^*. Concrete example of such a classification is given below.

Example 5: Suppose that the set of bottleneck links and the set of assignments are given as follows:

$$E^* = \{e_1, e_2, e_3\} \text{ and}$$
$$\mathcal{D} = \{(1,2,0), (2,1,0), (1,1,1), (0,2,1), (2,0,1)\}.$$

Then set $\mathcal{D}$ is classified into eight $(= 2^3)$ subsets as

$$\begin{aligned} \mathcal{D}_{\{e_1,e_2,e_3\}} &= \mathcal{D} \\ \mathcal{D}_{\{e_1,e_2\}} &= \{(1,2,0), (2,1,0)\} \\ \mathcal{D}_{\{e_2,e_3\}} &= \{(0,2,1)\} \\ \mathcal{D}_{\{e_1,e_3\}} &= \{(2,0,1)\} \text{ and} \end{aligned}$$

$\mathcal{D}_{E'} = \emptyset$ for every subset E' with size one or less, where the subscript of each subset indicates a subset of links supporting the assignments.

Let $r_{E'}$ be the reliability of G which is calculated by assuming that links in E' are available with probability 1 and links not in $E^* - E'$ are available with probability 0 (the way of calculating such reliabilities will be given soon). Then the reliability of G is calculated as

$$r(G) = \sum_{E' \subset E^*} p_{E'} \times r_{E'} \qquad (3)$$

where $p_{E'}$ is the probability of occurring the configuration E' of link failures (see Equation (2) for the details). Since the number of subsets of E^* is 2^k, we can calculate the above expression in constant time provided that $r_{E'}$ is given for each E' (recall that we are assuming that k is a constant which implies that 2^k is also a constant). Thus the remaining problem is *how to calculate $r_{E'}$ in $\mathcal{O}(2^{\alpha|E|})$ time*.

B. Reliability of G for Each Class of Assignments

In the proposed algorithm, the reliability $r_{E'}$ given in Equation (3) is calculated by accumulating the probabilities concerned with assignments in subset $\mathcal{D}_{E'}$, by using the inclusion-exclusion principle. Before describing the details of the accumulation algorithm, to help the understanding by the reader, we explain the flow of the algorithm using a simple example.

Example 6: Let $E' = \{e_1, e_2\}$ be a subset of bottleneck links and $\mathcal{D}_{E'} = \{b_1, b_2\}$ be the set of assignments supported by E'. Let $\mathcal{E}_s = \{c_1, c_2, c_3, c_4\}$ and $\mathcal{E}_t = \{c_5, c_6, c_7, c_8\}$ be sets of failure configurations in components G_s and G_t, respectively (in this example, we are assuming a very small component consisting of only two links; recall that $\mathcal{E}_s = 2^{E_s}$ and $\mathcal{E}_t = 2^{E_t}$). Suppose that each failure configuration in G_s and G_t realizes assignments of bottleneck links as shown in Table I; e.g., configuration c_1 for G_s realizes b_1, configuration c_5 for G_t realizes b_1 and b_2, and so on. Those patterns of realizations are recorded in the array-structured data, as was explained in Section III-C.

From this table, we can observe that the probability that G admits the given flow demand under assignment b_1 of bottleneck links is represented by the multiplication of: (a) the probability of occurring a failure configuration in G_s which realizes assignment b_1 and (b) the probability of occurring a failure configuration in G_t which realizes assignment b_1. Hence such a probability, denoted by $p_{\{b_1\}}$, is represented as

$$p_{\{b_1\}} = (p(c_1) + p(c_3)) \times (p(c_5) + p(c_7))$$

by assuming that the failure configuration designated by E' occurs with probability one, where $p(c_i)$ denotes the probability of occurring configuration c_i which can be calculated by the failure probability of the links. Similarly, the probability that G admits the flow demand under assignment b_2 is represented as

$$p_{\{b_2\}} = (p(c_2) + p(c_3) + p(c_4)) \times (p(c_5) + p(c_6)).$$

Thus, the probability such that G admits the given flow demand under assignment b_1 "or" b_2, which corresponds

TABLE I
ASSIGNMENTS REALIZED BY EACH FAILURE CONFIGURATION.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$\{b_1\}$	$\{b_2\}$	$\{b_1, b_2\}$	$\{b_2\}$	$\{b_1, b_2\}$	$\{b_2\}$	$\{b_1\}$	$\{\}$

to the required probability $r_{E'}$, is calculated as

$$r_{E'} = p_{\{b_1\}} + p_{\{b_2\}} - p_{\{b_1,b_2\}}$$

where $p_{\{b_1,b_2\}}$ is the probability that G admits the given flow demand under both assignments b_1 "and" b_2, which is calculated as

$$p_{\{b_1,b_2\}} = p(c_3) \times p(c_5)$$

in this example. The reader should note that this is a simple application of the inclusion-exclusion principle.

The above idea is summarized in the following algorithm for the accumulation.

pricedure ACCUMULATION

Input: Arrays for G_s and G_t recording the result of the calculation for subproblems; subset E' of E^*; and the set of assignments $\mathcal{D}_{E'}$ supported by E'.

Output: Reliability $r_{E'}$ of G under an assumption such that the failure configuration designated by E' occurs with probability one.

Step 1: For each subset $X \subseteq \mathcal{D}_{E'}$, calculate the probability p_X such that G admits the given flow demand under all assignments in X.

Step 2: By using all probabilities calculated in Step 1, calculate the probability $r_{E'}$ that G admits the given flow demand under at least one assignment in X by applying the inclusion-exclusion principle.

The execution time of the algorithm is evaluated as follows. Step 1 takes at most $2^{d^k} \times \max\{2^{|E_s|}, 2^{|E_t|}\}$ time since $\mathcal{D}_{E'} \subseteq \mathcal{D}$ and $|\mathcal{D}| \leq d^k$ (note that 2^{d^k} is still constant as long as d and k are constant !!). For the same reason, Step 2 takes at most 2^{d^k} time. Hence, since $\max\{|E_s|, |E_t|\} \leq \alpha|E|$ holds by assumption, we can conclude that if d and k are constant, we can complete the calculation of the reliability of G in $\mathcal{O}(2^{\alpha|E|})$ time.

V. CONCLUDING REMARKS

This paper proposes an algorithm for calculating the reliability of P2P video streaming systems with few bottleneck links. The proposed algorithm takes $\mathcal{O}(2^{\alpha|E|}|V||E|)$ time for calculating the reliability, where $\alpha|E|$ is the maximum number of links in connected components obtained by removing bottleneck links from G. The overview of two key procedures used in the proposed algorithm is illustrated in Figure 6.

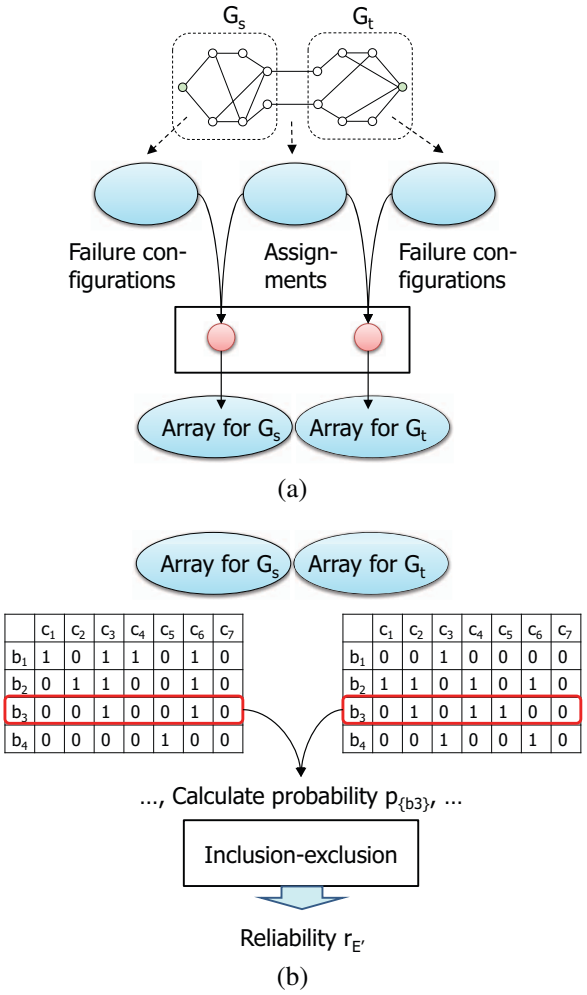

Fig. 6. Overview of two procedures proposed in this paper; (a) illustrates the procedure explained in Section III-C which generates two array-structures from sets of failure configurations and the set of assignments of flows to bottleneck links; (b) illustrates procedure ACCMULATION in Section IV-B.

REFERENCES

[1] B. A. Alghazawy and S. Fujita. "A Scheme for Maximal Resource Utilization in Peer-To-Peer Live Streaming." *International Journal of Computer Networks & Communications (IJCNC)*, 7(5): 13–28, 2015.

[2] G. Bianchi, N. B. Melazzi, L. Bracciale, F. L. Piccolo and S. Salsano. "Streamline: An Optimal Distribution Algorithm for Peer-to-Peer Real-Time Streaming." *IEEE Trans. on Parallel and Distributed Systems*, 21(6): 857–871, 2010.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh. "SplitStream: High-Bandwidth Multicast in Cooperative Environments." In *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP)*, 2003, pages 298–313.

[4] M. Castro, P. Druschel, A.-M Kermarrec and A. Rowstron. "SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure." *IEEE J.Sel. A. Commun.*, 20(8): 1489–1499, 2006.

[5] Y. Chu, S. G. Rao and H. Zhang. "A Case for End System Multicast." In *Proc. of the ACM Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2000, pages 1–12.

[6] G. Dan, V. Fodor and I. Chatzidrossos. "On the Performance of Multiple-Tree-Based Peer-to-Peer Live Streaming." In *Proc. of the 26th IEEE Int'l Conf. on Computer Communications (INFOCOM)*, 2007, pages 2556–2560.

[7] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms, Texts in Theoretical Computer Science. An EATCS Series*, Springer Berlin Heidelberg.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company.

[9] X. Jin, W.-P. K. Yiu, S.-H. G. Chan and Y. Wang. "On Maximizing Tree Bandwidth for Topology-Aware Peer-to-Peer Streaming." *IEEE Trans. on Multimedia*, 9(8):1580–1592, 2007.

[10] D. Kostić, A. Rodriguez, J. Albrecht and A. Vahdat. "Bullet: High Bandwidth Data Dissemination using an Overlay Mesh." In *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP)*, 2003, pages 282–297.

[11] Y. Liu. "On the Minimum Delay Peer-to-Peer Video Streaming: How Realtime Can It Be?" In *Proc. of the 15th ACM Int'l Conf. on Multimedia.*, 2007, pages 127–136.

[12] S. Liu, Z.-S. Rui, W. Jiang, J. Rexford and M. Chiang. "Performance Bounds for Peer-Assisted Live Streaming." In *Proc. of the ACM Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2008, pages 313–324.

[13] N. Magharei and R. Rejaie. "PRIME: Peer-to-Peer Receiver-Driven Mesh-Based Streaming." *IEEE/ACM Trans. on Networking*, 17(4): 1052–1065, 2009.

[14] V. Padmanabhan, H. Wang, P. Chou and K. Sripanidkulchai. "Distributing Streaming Media Content using Cooperative Networking." In *Proc. NOSSDAV'2*, 2002, pages 177–186.

[15] S. Ratnasamy, M. Handley, R. Karp and S. Shenker. "Application-Level Multicast Using Content-Addressable Networks." In *Proc. NGC'1*, 2001, pages14–29.

[16] F. Wang, Y. Xiong and J. Liu. "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming." *IEEE Trans. on Parallel and Distributed Systems*, 21(3): 379–392, 2010.

[17] X. Zhang, J. Liu, B. Li and Y. -S. P. Yum. "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming." In *Proc. the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies.* Vol. 3, 2005, pages 2102–2111.

[18] M. Zhang, Q. Zhang, L. Sun and S. Yang, "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?," *IEEE J.Sel. A. Commun.*, 25(9): 1678–1694, 2007.

[19] Y. Zhao, Y. Liu, C. Chen and J. Zhang. "Enabling P2P One-View Multiparty Video Conferencing." *IEEE Trans. on Parallel and Distributed Systems*, 25(1): 73–82, 2014.

[20] S. Zhuang, B. Zhao, A. Joseph, R. Katz and J. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination." In *Proc. NOSSDAV'1*, 2001, pages 11–20.