

Achieving Maximal Path Diversity in Low-Delay Application-Level Multicast

Maciej Kurant, Patrick Thiran

LCA - School of Communications and Computer Science
EPFL, CH-1015 Lausanne, Switzerland

Abstract—We consider the Application-Level Multicast (ALM) for interactive applications. Such a system should (i) respect stringent delay requirements, and proactively protect the system against (ii) overlay node failures and (iii) the packet losses at the IP layer. We achieve this as follows.

First, we define a metric called Vulnerability \mathcal{W} , based purely on the topology of the ALM system. We prove that, under some assumptions, \mathcal{W} is equivalent to the average packet loss rate observed at the destinations. This crucial observation allows us to focus directly on minimizing \mathcal{W} , which significantly simplifies the problem.

Second, we note that \mathcal{W} drops with the amount of IP-level and overlay-level path diversity available in the system. Therefore, we consider typical techniques to create a number of alternative paths in ALM, such as adding redundant cross-links, or using a set of multiple distribution trees. We propose a framework that accommodates and generalizes these approaches.

Third, within this framework we optimize the structure of ALM. In simulations on real Internet topologies, we find that maximizing the overlay-path diversity (that roughly represents the state of the art) may result in a poor IP-path diversity. Therefore, we propose to include the IP topology in the objective function and to maximize both the overlay-level and IP-level path diversity at the same time. As a result, we proactively protect the ALM system against overlay node failures and IP losses, reducing its Vulnerability \mathcal{W} (and thus the effective loss rate) by typically 30%-70%. Moreover, we develop a set of lower-bounds on \mathcal{W} and show that our approach is nearly optimal. Finally, we study factors that naturally limit the available IP-path diversity, such as the system size and maximal allowed redundancy.

I. INTRODUCTION AND RELATED WORK

Application-level multicast (ALM) has been a hot topic in the last decade, which has resulted in numerous protocols and systems, e.g., Chaining [1], Narada [2], Nice [3], HMTF [4], SplitStream [5], ZigZag [6], OMNI [7], Coop-Net [8,9], Promise [10], CoolStreaming [11], Active [12], AnySee [13], Prime [14], PPLive [15]. These solutions can be roughly divided into two groups: regular ALMs and interactive ALMs.

Regular ALM, or P2P streaming, focuses on the delivery of a high-quality data stream (e.g., a TV channel) from a single source to a large number of destinations. As this typically involves a one-way communication, the acceptable time lag is in the order of seconds or even minutes [14,15]. Consequently, when a node leaves the system, all nodes connected to it have enough time to find another peer with the context, without pausing the stream playout. In other words, regular ALM can be effectively protected by *reactive* failure recovery techniques [6,16]. Moreover, some systems rely on pull-based techniques also during the regular dissemination

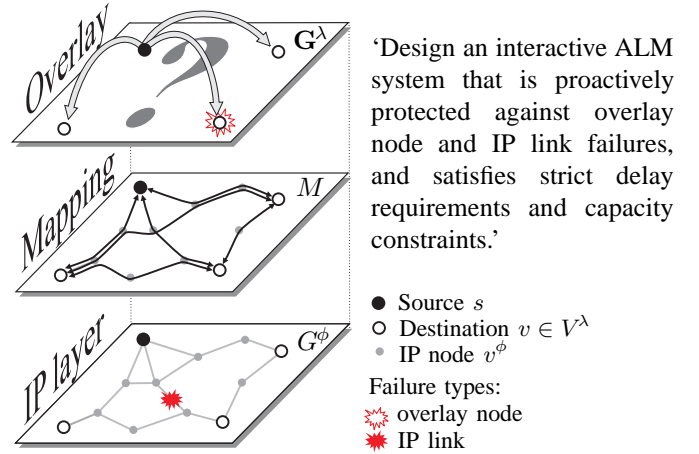


Fig. 1. General problem illustration.

of the stream [11,14,15], which is a scalable and resource-efficient approach [17].

In contrast, *interactive ALM* addresses applications such as tele- and video-conference [12,16] or network multiplayer games [18,19]. These settings have specific features, requirements and design challenges [16] very different from those of regular ALM, as follows.

First, interactive ALM has very *strict delay requirements*, typically not exceeding a few hundreds of milliseconds. Therefore, it cannot rely on pull-based data distribution mechanisms, as they result in unacceptable delays. Instead, the stream should be quickly *pushed* through an organized structure such as a delay-optimized tree [7,16,20,21] for an overview).

Second, even if the total number of participants of a teleconference is large, usually only a small group of them interact directly [12,16,22]. Similarly, in network multiplayer games, the players can be organized into small groups of interacting users who are close to each other in the virtual world [18]. Therefore, interactive applications are typically of a *limited size*, ranging from a few to several tens of nodes.

Third, reactive failure recovery schemes usually take too much time to meet the target delay requirements of interactive applications [23]. Instead, we should use *proactive failure protection techniques* that guarantee timely data delivery in the presence of most common failures.

Finally, the main concern in regular ALM are *overlay node failures* due to the node departures (i.e., ‘node churn’). In interactive ALM, however, this problem is naturally limited, due to the small number of peers and their commitment to the application. In contrast, what can become a serious

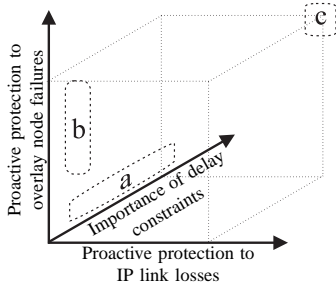


Fig. 2. Existing ALM approaches (a,b) and the goal of this work (c) with respect to three factors critical in interactive ALM.

impediment are *packet losses at the IP layer*. Indeed, a lost IP packet can be easily and automatically retransmitted at the cost of larger delays allowed in regular ALM, but it must be taken care of separately under the low-latency constraint.

To conclude, an interactive ALM system should (i) push the data stream through a delay-constrained structure, and proactively protect the system against (ii) overlay node failures and (iii) IP link losses (see illustration in Fig. 1).

This problem has been only partially addressed to date. For example, minimizing delays is the main goal in [7,12,16,20]. As these solutions typically use failure protection that is reactive, they fall in the category (a) illustrated in Fig. 2.

Another group of approaches, indicated by (b) in Fig. 2, are the ALM systems that exploit the *overlay path diversity* combined with some redundancy codings [5,9,24]. This results in a good protection to overlay node failures and some (limited) protection to IP losses, but the strict delay constraint is not among the primary concerns.

The objective of this paper is to satisfy simultaneously all three goals (i), (ii) and (iii), which places us in the region (c) in Fig. 2. In order to achieve this, we leverage not only on the overlay path diversity, but also on the *IP path diversity*. Indeed, the existence of multiple IP paths was successfully exploited to fight IP losses in the context of the unicast delay-constrained connection [25]–[27]. We find that if we take the underlying IP layer explicitly into account, we can usually maximize *both* the overlay-level and IP-level path diversity at the same time. As a result we proactively protect the ALM against overlay node failures *and* IP losses. Moreover, our solution respects the strict delay constraints and uses only the available resources (i.e., nodes participating in the application, and no ‘exterior’ relay nodes) and protocols (i.e., no source routing).

Taking the topology of the underlying IP layer into account proved to be effective also in related scenarios, such as placing new overlay nodes within an ISP [28], choosing a good overlay backup path [29]–[31], streaming from several sources (peers) to one receiver [10], or overlay routing [32,33].

We formulate the problem in Section II. Next, in Section III we present a simple topology-based version of the problem, and prove that under some conditions the two formulations are equivalent. In Sections IV–VI, we discuss the techniques, objective functions and heuristics that introduce path diversity and redundancy in the system, leading to good solutions of

ALM s, V^λ $G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$	Application-Layer Multicast source node, destination nodes directed overlay graph with all possible overlay edges E^λ
$\gamma_{max} \geq 0$	maximal redundancy
$t_{max}(v)$	maximal allowed delivery time
$b_{max}(v)$	maximal uplink bandwidth ratio
$M(e^\lambda)$	IP mapping of overlay link e^λ
E^ϕ	set of all IP links in the system
$C^\lambda(v), C^\phi(v)$	set of critical overlay nodes / IP links of node v
$\mathcal{W}^\lambda, \mathcal{W}^\phi$	logical and physical vulnerability
$\mathcal{W}(\eta) = \eta\mathcal{W}^\lambda + \bar{\eta}\mathcal{W}^\phi$	total vulnerability

TABLE I
BASIC NOTATION USED IN THIS PAPER.

our problem. We evaluate our approach on real-life Internet topologies in Section VII. Finally, in Section VIII we conclude the paper.

II. NOTATION AND PROBLEM FORMULATION

LONG

We assume that there is one stream source s (e.g., a lecturer). The other nodes should get the stream reliably and within the delay deadline. The existence of one main source of stream is typical of many interactive applications such as a teleconference [16]. This scenario can also be considered as a basic building block of a reliable all-to-all multicast system.

A. Overlay layer and capacity constraints

Let s be the traffic *source* node and let V^λ be a set of *destination nodes*, illustrated in Fig. 1. The source s generates *source packets* destined to all nodes in V^λ . The source s can send the packets to any destination node(s) that may forward them to any other destination node(s); each such communication link is interpreted as a directed overlay edge $e^\lambda \in E^\lambda$. Denote by $G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$ the directed graph with all possible overlay edges E^λ , $|E^\lambda| = |V^\lambda| + |V^\lambda|(|V^\lambda| - 1) = |V^\lambda|^2$. In practice, we cannot usually use all the edges in E^λ at the same time because of the following capacity constraints.

First, every node v has limited downlink and uplink capacities. The latter is a critical factor in the construction of ALM [34]. Therefore, we denote by $b_{max}(v)$ the maximal outgoing bandwidth of node v normalized by the data stream rate. For example, $b_{max}(v) = 2.5$ when the uplink capacity of v is 1Mbit/s and the stream rate is 0.4Mbit/s. We refer to this system-specific feature as *local capacity constraint*.

Second, *global capacity constraint* γ_{max} is the maximal amount of redundancy we are allowed to introduce in the entire system. For example, $\gamma_{max} = 0.2$ means that we can add up to 20% of redundant traffic on top of the primary traffic (equal to $|V^\lambda| \cdot stream_rate$). This constraint is set (or relaxed) by the application. Fixing γ_{max} allows us to make a fair comparison between various protection schemes.

B. IP layer

The overlay is built on top of the underlying IP layer (see Fig. 1). To avoid confusion, we use ‘ ϕ ’ (‘physical’) to denote IP elements and ‘ λ ’ (‘logical’) for overlay elements. Every

overlay edge $e^\lambda = (v, u)$ in E^λ defines a unicast path in the underlying IP layer. It is a sequence of IP links $(e_1^\phi, e_2^\phi, \dots)$ that can be discovered by running a traceroute from v to u . We call this path a *mapping* of edge e^λ and denote by $M(e^\lambda)$ or $M(v, u)$. Finally, E^ϕ is the set of all IP links that can possibly be used by our overlay, i.e., $E^\phi = \bigcup_{e^\lambda \in E^\lambda} M(e^\lambda)$.

Clearly, we have no control over the IP layer topology G^ϕ and over the mapping M , which we therefore consider as fixed and given. In contrast, we have full control over the overlay layer.

C. Packet losses

We consider overlay node failures and IP link failures. An *overlay node failure* is usually a departure of a node $v \in V^\lambda$ from the application, with no prior notification. Clearly all packets that were scheduled to be forwarded by v are lost until the system adapts to the new situation.

An *IP link failure* is usually caused by traffic congestion, which results in the loss of one or a burst of packets [35]. As these problems are typically short-lived, they are often classified as ‘soft failures’, to stress the difference from ‘hard failures’ (e.g., a cut of the IP link); we use the single-word term ‘failure’ for simplicity.

D. Maximal allowed time $t_{max}(v)$ and average loss rate \bar{r}

We say that a source packet i is *delivered* at a destination v when i (or its copy) reaches v or i is reconstructed at v from other packets that were successfully delivered. Let $t_{max}(v)$ be the maximal delay acceptable (by the application) between the creation of a source packet and its delivery at node $v \in V^\lambda$. This delay is composed purely of the propagation times at the IP layer - the processing delays at the overlay nodes are ignored as they are very small (less than 1ms) for non-overloaded nodes and when high priority forwarding is used [36]. $t_{max}(v)$ may be the same for all nodes, or not. Denote by $r(v)$ the source packet loss rate observed at node v , i.e., the probability that a source packet is *not* delivered at v within time $t_{max}(v)$. Finally, $\bar{r} = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} r(v)$ is the source packet loss rate averaged over all traffic destinations.

E. General problem formulation: Problem P1

Now we can state the general problem as follows:

P1: *Given capacity and delay constraints $\{b_{max}(v), \gamma, t_{max}(v)\}$ for $v \in V \cup \{s\}$, design an ALM system that minimizes the average source packet loss rate \bar{r} under the presence of overlay node failures and IP link losses.*

In other words, we maximize the robustness of ALM to overlay node and IP link failures, while satisfying strict delay requirements and capacity constraints (see Fig. 1).

III. TOPOLOGY-BASED FORMULATION: PROBLEM P2

Our primary goal is to solve P1. However, to evaluate the average source packet loss rate \bar{r} , we have to specify not only the capacity and delay constraints, but also the packet loss model together with all loss parameters for every overlay node

and IP link. This quickly gets very complicated and dependent on a myriad of heterogenous elements.

For this reason, in this section we define a problem P2 that is based purely on the *topological* aspects of the system. P2 unifies all these elements under one homogenous framework, making it much easier to handle. Moreover, we formally link the two problems - we show that, under some conditions, P1 and P2 are equivalent.

A. Problem P2

Let us begin with the following definition:

Definition 1 (Critical components $C^\lambda(v)$ and $C^\phi(v)$): We say that a network component $x \in E^\phi \cup V^\lambda \setminus \{v\}$ is *critical* for an overlay node $v \in V^\lambda$, if during a single and permanent failure of x , the source packets are not delivered at v within time $t_{max}(v)$. Denote by $C^\lambda(v) \subset V^\lambda \setminus \{v\}$ the set of all critical overlay nodes of v , and by $C^\phi(v) \subseteq E^\phi$ the set of all critical IP links of v .

Consider the example in Fig. 3a. The sets of critical nodes of v and u are, respectively, $C^\lambda(v) = \{u\}$ and $C^\lambda(u) = \emptyset$, respectively. (Note that by definition, source s is not included in these sets.) Similarly, the sets of critical IP links of these nodes are $C^\phi(v) = \{a^\phi, b^\phi, c^\phi, d^\phi, e^\phi\}$ and $C^\phi(u) = \{a^\phi, b^\phi, c^\phi\}$, respectively. However, adding one more active link to the overlay topology as shown in Fig. 3b greatly changes the situation of node v . Indeed, now no single overlay node failure or physical link failure can disconnect v from s and thus $C^\lambda(v) = C^\phi(v) = \emptyset$.

We can now define the metric that captures the system vulnerability to single physical and overlay failures.

Definition 2 (Vulnerability \mathcal{W}^λ , \mathcal{W}^ϕ and \mathcal{W}): We define the logical Vulnerability \mathcal{W}^λ (to overlay node failures) and the physical Vulnerability \mathcal{W}^ϕ (to IP link failures) as

$$\mathcal{W}^\lambda = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} |C^\lambda(v)|, \quad \mathcal{W}^\phi = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} |C^\phi(v)|.$$

We average these two vulnerabilities in a single metric \mathcal{W} ,

$$\mathcal{W}(\eta) = \eta \cdot \mathcal{W}^\lambda + (1 - \eta) \cdot \mathcal{W}^\phi, \quad (1)$$

for some averaging coefficient $0 \leq \eta \leq 1$ that weights the relative contributions of \mathcal{W}^λ and \mathcal{W}^ϕ .

Vulnerabilities \mathcal{W}^λ and \mathcal{W}^ϕ are determined by the topology of the ALM. For example, in Fig. 3a, we have $\mathcal{W}^\lambda = 1/|V^\lambda|$ (contributed by node v) and $\mathcal{W}^\phi = (3+3+5)/|V^\lambda| = 11/|V^\lambda|$. Adding one active overlay link (as in Fig. 3b) creates an alternative path between s and v , which decreases vulnerabilities to $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = 6/|V^\lambda|$, respectively. This also hints that \mathcal{W}^λ and \mathcal{W}^ϕ are not independent.

Vulnerability metrics capture the path diversity in the system, which can be interpreted as follows:

$$\begin{aligned} \text{Small } \mathcal{W}^\lambda &\Leftrightarrow \text{High overlay-path diversity.} \\ \text{Small } \mathcal{W}^\phi &\Leftrightarrow \text{High IP-path diversity.} \end{aligned}$$

Vulnerability $\mathcal{W}(\eta)$ is designed to jointly capture the IP losses and overlay node failures, by weighting \mathcal{W}^λ and \mathcal{W}^ϕ with a

parameter η . Therefore, we define the Problem P2 as follows:

P2: *Given capacity and delay constraints $\{b_{max}(v), \gamma, t_{max}(v)\}$ for $v \in V \cup \{s\}$, design an ALM system that minimizes Vulnerability $\mathcal{W}(\eta)$ for a given parameter η .*

B. Equivalence of P1 and P2

In order to link P1 and P2, we make two assumptions on the loss model, A1 and A2. They can be considered as first order approximations of the reality. First, as we have no prior knowledge of the failure probabilities of particular components, we assume that

A1: *Every overlay node $v \in V^\lambda$ fails with the same probability $0 \leq p^\lambda < 1$, independently of other elements. Every IP link $e^\phi \in E^\phi$ fails with the same probability $0 \leq p^\phi < 1$, independently of other network components.*

Second, we assume that

A2: *Failures are persistent, i.e., a source packet i and all its duplicates and derivatives observe the same state of every network component.*

This is a natural assumption for overlay node failures, as they are typically long-lasting (e.g., leaving the system). Moreover, A2 roughly captures the bursty nature of losses at IP links in today's Internet [35]. Indeed, A2 assumes ideal, 'all or none' type of bursts. This simplifies the aspects of time variability and allows us to focus explicitly on achieving high path diversity, which is our main goal.

Now we can state the following theorem:

Theorem 1 (Equivalence of P1 and P2):

Given Assumptions A1-A2, we have that

$$\bar{r} = \alpha \cdot \mathcal{W}(\eta) + O((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2), \quad (2)$$

for a system-specific constant

$$\alpha = (p^\lambda + p^\phi - 2p^\lambda p^\phi) \cdot (1 - p^\lambda)^{|V^\lambda|-1} (1 - p^\phi)^{|E^\phi|-1}$$

and the parameter η equal to

$$\eta = \frac{p^\lambda (1 - p^\phi)}{p^\lambda + p^\phi - 2p^\lambda p^\phi}. \quad (3)$$

Proof: See Appendix.

As the overall failure probability in the Internet is very low (implying small p^ϕ) and the interactive ALMs are usually small, stable systems (implying small $|E^\phi|$, $|V^\lambda|$ and p^λ), the term $O((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2)$ is negligible in practice. Indeed, this term captures the probability of having multiple failures; if we additionally assume that at most one failure occurs at a time, then (2) boils down to precisely $\bar{r} = \alpha' \cdot \mathcal{W}$.

An immediate corollary of Theorem 1 is that, given A1-A2, minimizing the average loss rate \bar{r} is (almost) equivalent to minimizing Vulnerability $\mathcal{W}(\eta)$ with η set as shown in (3). Therefore, from now on we focus explicitly on Vulnerability and on solving P2 rather than P1.

IV. PROTECTION TECHNIQUES

So far we have defined a meaningful metric $\mathcal{W}(\eta)$ that we want to minimize, but we have not said how to achieve this in practice. In this section we discuss four protection techniques that allow us to introduce the path diversity and redundancy in the system, and, as a result, to optimize $\mathcal{W}(\eta)$. Please refer to Fig. 3 for examples of each technique.

A. SingleTree (Fig. 3a)

The first technique is only used as a reference point with strictly *no* redundancy ($\gamma = 0$), i.e., a directed tree $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$, $|E_1^\lambda| = |V^\lambda|$, rooted at the source s . This straightforward ALM solution was used e.g., in [2]–[4,6,7,12,16,20]. Check [21] for a review and comparison. Most of these papers propose some heuristics to minimize the average or maximal end-to-end propagation time, as this problem is in general NP-complete [7,20]. We refer to this basic technique as *SingleTree*.

B. SingleGraph (Fig. 3b)

The obvious problem with *SingleTree* is that it does not provide any proactive protection - any single failure always affects all nodes located downstream of the failing element. One way of increasing its resilience is to use a distribution graph $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$ with *more edges than necessary*, i.e., with $|E_1^\lambda| > |V^\lambda|$. All edges E_1^λ actively forward all the packets resulting in a number of alternative paths leading to destination nodes. The redundancy of this system is $\gamma = (|E_1^\lambda| - |V^\lambda|)/|V^\lambda| = R/|V^\lambda|$, where $R = |E_1^\lambda| - |V^\lambda|$ is the number of redundancy edges, i.e., the maximal number of edges without which the necessary traffic can be still delivered.

How can we construct a good graph G_1^λ ? One method is an incremental design starting from a *SingleTree* topology, to which a number of 'cross-edges' are added, as e.g., in PRM [24], TMesh [37], HBM [38] and in [39]. Instead, we propose to construct G_1^λ starting from scratch, which clearly gives us more freedom and potentially leads to a better performance. We call this general approach *SingleGraph*.

C. MultiTree (Fig. 3c)

Another technique to introduce redundancy in the system is coding. In particular, we consider a non-systematic Forward Error Correction FEC(n, k) scheme where each source packet is encoded as one FEC block of n packets, called *FEC* packets. The total size of the FEC block is n/k times larger than the original source packet, so the introduced redundancy is $\gamma = (n - k)/k$. If k or more FEC packets are received within time $t_{max}(v)$ at destination v , then the original source packet is reconstructed and delivered; otherwise it is lost. In order to benefit from the available path diversity, we construct a set $\{G_1^\lambda, \dots, G_n^\lambda\}$ of n trees, and disseminate the i th FEC packet over the i th tree G_i^λ . We refer to this technique as *MultiTree*.

Multiple trees were first used together in ALM systems such as CoopNet [8,9] and SplitStream [5]. These systems use Multiple Description Coding (MDC) instead of FEC. But they share the same general principle - exploiting path diversity by disseminating encoded packets over different trees.

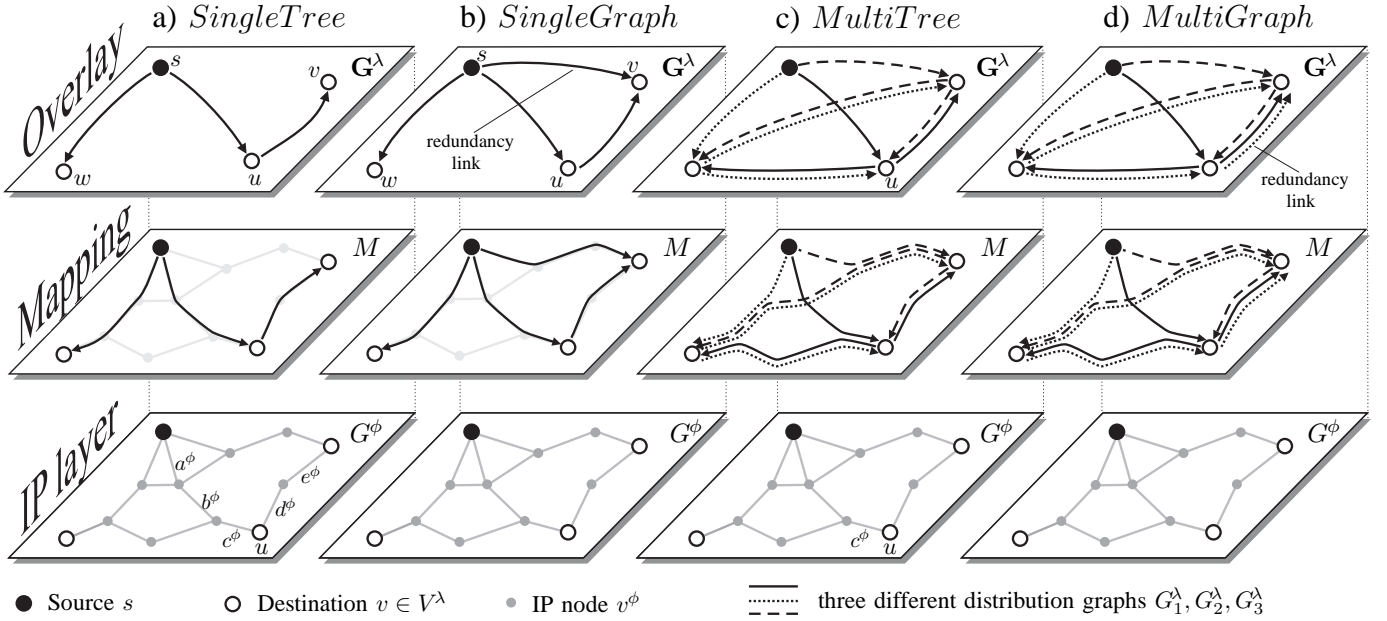


Fig. 3. An illustration of four protection techniques in a system with one source and $|V^\lambda|=3$ destinations. To simplify the presentation we set $t_{max} = \infty$. (a) *SingleTree*: $\gamma = 0$ (no redundancy, reference point) and $\mathcal{W}^\lambda = \frac{1}{|V^\lambda|}$ and $\mathcal{W}^\phi = \frac{11}{|V^\lambda|}$. (b) *SingleGraph*: $\gamma = (4-3)/3 = 1/3$ ($R = 1$ redundancy link), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{6}{|V^\lambda|}$. (c) *MultiTree* with FEC(3,2): $\gamma = (3-2)/2 = 1/2$ (only FEC redundancy), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{5}{|V^\lambda|}$. (d) *MultiGraph* with FEC(3,2): $\gamma = 1/2 + 1/9$ (FEC redundancy plus $R = 1$ redundancy link in one distribution graph), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{3}{|V^\lambda|}$.

D. MultiGraph (Fig. 3d)

Finally, we propose to combine *MultiTree* and *SingleGraph* by making at least one of the distribution graphs denser than a tree, i.e., $n > 1$ and $|E_i^\lambda| > |V^\lambda|$ for at least one $i, 1 \leq i \leq n$.¹ Under this technique, called *MultiGraph*, the number of edges needed to carry the necessary traffic (with no redundancy) is equal to $k \cdot |V^\lambda|$ (k trees). So every edge in the system carries fraction $1/(k \cdot |V^\lambda|)$ of the necessary traffic. Thus the system redundancy γ is

$$\gamma = \frac{\sum_{i=1}^n |E_i^\lambda|}{k \cdot |V^\lambda|} - 1 = \frac{n-k}{k} + \frac{R}{k \cdot |V^\lambda|},$$

where R is the total number of redundancy edges in all distribution graphs, $R = \sum_{i=1}^n (|E_i^\lambda| - |V^\lambda|)$.

To the best of our knowledge, no technique equivalent to *MultiGraph* has been studied to date. In the Appendix we demonstrate on a concrete example the following:

Observation 1 (Each protection technique may be best):

Consider a source s , a set V^λ of destinations and some fixed redundancy $\gamma_{max} > 0$. Depending on the underlying IP topology, each of the three protection techniques (*SingleGraph*, *MultiTree* and *MultiGraph*) may alone lead to the smallest vulnerability \mathcal{W}^ϕ .

E. A common framework \mathbf{G}^λ

Let us now introduce a common notation that accommodates all the above redundancy techniques. By convention, we say that our system always uses FEC(n, k) with $n \geq k \geq 1$. Denote by $\mathbf{G}^\lambda = \{G_1^\lambda, \dots, G_n^\lambda\}$ a set of n dissemination

¹Note that according to this definition neither *SingleGraph* nor *MultiTree* is a *MultiGraph*.

Technique	n and k	number of edges	redundancy γ
<i>SingleTree</i>	$n=k=1$	$ E_1^\lambda = V^\lambda $	0
<i>SingleGraph</i>	$n=k=1$	$ E_1^\lambda > V^\lambda $	$\frac{ E_1^\lambda - V^\lambda }{ V^\lambda }$
<i>MultiTree</i>	$n > k \geq 1$	$ E_i^\lambda = V^\lambda $ for all $1 \leq i \leq n$	$\frac{n-k}{k}$
<i>MultiGraph</i>	$n > k \geq 1$	$ E_i^\lambda > V^\lambda $ for at least one i	$\frac{\sum_{i=1}^n E_i^\lambda }{k \cdot V^\lambda } - 1$

TABLE II
FOUR PROTECTION TECHNIQUES WITHIN THE \mathbf{G}^λ FRAMEWORK.

overlay graphs $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda \subseteq E^\lambda)$, $|E_i^\lambda| \geq |V^\lambda|$ for every $1 \leq i \leq n$. We show in Table II that all four protection techniques described above are some special cases of the general framework \mathbf{G}^λ .

Feasibility. We say that the set \mathbf{G}^λ is *feasible* if both local and global capacity constraints are satisfied, and if in every graph G_i^λ , every node $v \in V^\lambda$ is reachable from s within time $t_{max}(v)$. Naturally, for every protection technique we require that it yields a feasible topology \mathbf{G}^λ .

Critical components. In order to evaluate Vulnerability $\mathcal{W}(\eta)$, we need a way to assess if a network component is critical in the system (see Def. 1,2). In our framework \mathbf{G}^λ , it is simple and well defined. To achieve this, we will say that a network component x (overlay node or IP link) is critical for a node $v \in V^\lambda$ in a separate distribution graph $G_i^\lambda \in \mathbf{G}^\lambda$ if every path from s to v in G_i^λ shorter than $t_{max}(v)$ traverses x . Now, a network component x is critical for $v \in V^\lambda$ in the entire system \mathbf{G}^λ if x is critical for v in more than $n-k$ distribution graphs G_i^λ . Indeed, during the failure of such a component x , more than $n-k$ FEC packets do not reach v making the FEC recovery impossible.

For example, in Fig. 3c, the IP link e^ϕ is critical for node u in two distribution graphs: G_1^λ (plain lines) and G_2^λ (dotted

lines). As $2 > n - k = 1$, we know that c^ϕ is critical for u in the entire system \mathbf{G}^λ , i.e., $c^\phi \in C^\phi(u)$.

V. OBJECTIVE FUNCTIONS

Recall that our goal is solving the problem P2, i.e., finding a feasible topology \mathbf{G}^λ that minimizes the vulnerability $\mathcal{W}(\eta)$. To the best of our knowledge, this general problem has not been addressed to date. Therefore, there is no prior work that we can directly compare with. However, many works address simplified versions of P2, which amount to replace $\mathcal{W}(\eta)$ by another function X . We distinguish three general categories of these objective functions: RAND, 1-LAYER and 2-LAYER, as follows.

A. RAND (reference point)

Under RAND, we select the topology of \mathbf{G}^λ at random, only guaranteeing its feasibility. Other than that, no effort is made to minimize $\mathcal{W}(\eta)$. This simple approach was used together with *SingleGraph* e.g., in PRM [24] where the additional cross-links were randomly added to the distribution tree. In the context of *MultiTree* the examples are CoopNet [8], and the RMF scheme in [40]. RAND also serves as a reference point for other solutions.

B. 1-LAYER (state of the art)

In contrast, 1-LAYER optimizes the overlay topology \mathbf{G}^λ , but *ignores the knowledge of the underlying IP layer*. Therefore, under 1-LAYER we directly minimize the logical vulnerability \mathcal{W}^λ only. This should result in some level of IP path diversity too, but we have no direct control over the physical vulnerability \mathcal{W}^ϕ . The 1-LAYER objective function captures many existing solutions. Under *SingleGraph*, maximizing the overlay path diversity is one of the main goals in [38,39,41]. Similarly, for *MultiTree* a set of node-disjoint trees is constructed in SplitStream [5], and in [9,40]. Thus 1-LAYER corresponds to the state of the art, with the addition of a strict delay constraint that we consider here, contrary to the works mentioned above.

C. 2-LAYER (our proposal)

Finally, under 2-LAYER we propose to take *both* the overlay and the IP layer into account. This additional information allows us to directly minimize the entire Vulnerability $\mathcal{W}(\eta)$ for any $0 \leq \eta \leq 1$. To the best of our knowledge no ALM technique that falls in this category has been proposed to date. One of the goals of this paper is to study the gain of 2-LAYER over 1-LAYER.

To conclude, within a given protection technique (*SingleGraph*, *MultiTree*, *MultiGraph*), we minimize

$$X = \begin{cases} 0 & \text{under RAND} \\ \mathcal{W}^\lambda & \text{under 1-LAYER} \\ \mathcal{W} & \text{under 2-LAYER} \end{cases}, \quad (4)$$

subject to the feasibility of \mathbf{G}^λ .

VI. CONSTRUCTING A GOOD TOPOLOGY \mathbf{G}^λ

In the previous two sections we specified our protection techniques and objective functions. The last step is to find the actual topology \mathbf{G}^λ that minimizes the objective function (4) under one of the protection techniques described in Section IV. Unfortunately, finding the optimal \mathbf{G}^λ is an NP-complete problem. Indeed, already finding a feasible topology \mathbf{G}^λ with $n = k = 1$, $\gamma = 0$ and fixed $b_{max}(v)$ for some t_{max} is equivalent to the ‘minimum maximum-latency degree-bounded directed spanning tree problem’ [7,42] shown to be NP-complete.

The problem complexity makes the computation of the optimal solutions untractable for more than a few nodes. Therefore, we use a *simulated annealing* heuristic to optimize the topology of \mathbf{G}^λ . The *input data* are the full graph $G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$ with propagation delays for every link, the mapping M , FEC parameters n, k , local and global capacity constraints and maximal delays $t_{max}(v)$. As the *initial topology* of \mathbf{G}^λ we take a set of n different connected random graphs with the total number of edges equal to

$$\sum_{i=1}^n |E_i^\lambda| = \lfloor |V^\lambda| \cdot (1 + \gamma_{max} - \frac{n-k}{k}) \rfloor. \quad (5)$$

This guarantees that the global capacity constraint γ_{max} is satisfied and maximally exploited. At every iteration we add, delete or rewire one or more edges, always preserving (5). In particular, we allow for the following topology change operations on \mathbf{G}^λ :

- 1) ‘Rewire source’: in one of the distribution graphs G_i^λ change a randomly chosen overlay edge $e^\lambda = (v_1, w)$ into $e^\lambda \leftarrow (v_2, w)$, where v_2 is picked at random.
- 2) ‘Rewire target’: like ‘Rewire Source’, but changes $e^\lambda = (v, w_1)$ into $e^\lambda \leftarrow (v, w_2)$.
- 3) ‘Rewire pair’: change a randomly chosen overlay pair of edges $e_1^\lambda = (v_1, w_1)$ and $e_2^\lambda = (v_2, w_2)$ into $e_1^\lambda \leftarrow (v_1, w_2)$ and $e_2^\lambda \leftarrow (v_2, w_1)$.

The heuristic minimizes the value of X (defined in (4)) and returns the topology \mathbf{G}^λ corresponding to the smallest found X .

In the Appendix we describe two techniques we used to significantly speed-up the heuristic. First, we propose a Bellman-Ford-based algorithm that efficiently finds the critical components and calculates Vulnerability \mathcal{W} . Second, we give a useful theorem that allows us to evaluate the effect of a topology change without actually recalculating \mathcal{W} .

A. Lower-bounds on \mathcal{W}^λ and \mathcal{W}^ϕ

In order to evaluate the performance of this heuristic and get some insight into the problem, we derive a number of lower-bounds on vulnerabilities \mathcal{W}^λ and \mathcal{W}^ϕ . The first one is called *CompleteGraph*; it is a simple, general and rather conservative lower-bound on \mathcal{W}^ϕ . The remaining bounds are specifically dedicated to given protection techniques. In particular, we derive *SingleGraph/SingleTree* lower-bounds on \mathcal{W}^λ and \mathcal{W}^ϕ , and a *MultiTree* lower-bound on \mathcal{W}^ϕ .

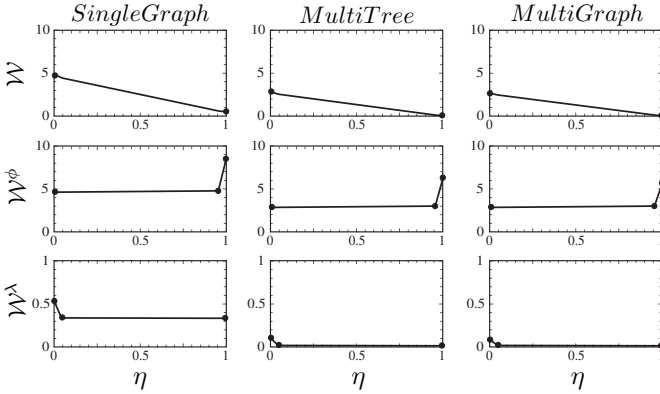


Fig. 4. Vulnerability $\mathcal{W}(\eta)$ under 2-LAYER, as a function of the weighting parameter η (top). Below we show physical Vulnerability \mathcal{W}^ϕ (middle) and overlay Vulnerability \mathcal{W}^λ (bottom) of the resulting topology \mathbf{G}^λ . We simulated $\eta \in \{0, 0.05, 0.95, 1\}$, averaged over 1000 random subsets of the DIMES data set with $|V^\lambda|=10$, $\gamma_{max}=0.5$ and $b_{max}(v)=2$.

As these lower bounds are tedious to derive, we moved their presentation to the Appendix. However, they lead us to important observations that we describe in the next section.

VII. SIMULATION RESULTS

A. Data sets

We collected the all-to-all traceroutes between 800 nodes participating in the DIMES project [43]. These nodes are private users scattered around the world, who voluntarily run the DIMES measurement software in the background. In order to obtain a dense and representative data set, we kept only those nodes that were active throughout the entire experiment and have different geographical locations. We also discarded the unsuccessful traceroutes (that did not reach the destination or have some unknown hops) and excluded the access links from the remaining traceroutes. The resulting data set consists of 107 nodes and about 90% of all 107×106 possible traceroutes.

We also conducted an analogous set of experiments on PlanetLab [44] and obtained results similar to those of DIMES.

B. Setting t_{max}

The minimal propagation delays that the system can achieve strongly depend on the choice of source s and destinations V^λ . Therefore, fixing t_{max} across all simulation runs would be difficult to interpret. Instead, we decided to choose t_{max} for every set $V^\lambda \cup \{s\}$ separately, by allowing for slightly more time than strictly necessary. In particular, we use the CPT centralized heuristic [20] to find the (approximated) propagation delay t_{min} to the farthest destination in the delay optimized *SingleTree*. If not stated otherwise, we set $t_{max}(v) = t_{min} + 50ms$ for every destination v .

C. Vulnerability $\mathcal{W}(\eta)$ and topology \mathbf{G}^λ as a function of η .

Our first goal is to understand how different values of the weighting parameter η affect the optimized topology \mathbf{G}^λ . This study does not concern RAND and 1-LAYER, because, according to (4), these two objective functions are independent of η . In contrast, under 2-LAYER, the objective is to minimize

the vulnerability $\mathcal{W}(\eta) = \eta\mathcal{W}^\lambda + (1-\eta)\mathcal{W}^\phi$, which clearly depends on η .

In Fig. 4 we study $\mathcal{W}(\eta)$ minimized by our heuristic, as a function of η . We also present the vulnerabilities \mathcal{W}^ϕ and \mathcal{W}^λ of the resulting topology \mathbf{G}^λ , which leads us to the first conclusion C1:

C1: For a wide range $0 < \eta < 1$ (but not for $\eta = 0$ nor $\eta = 1$) under 2-LAYER, both \mathcal{W}^λ and \mathcal{W}^ϕ are minimized at the same time.

To support C1, we consider three cases: $\eta = 0$, $\eta = 1$ and $0 < \eta < 1$. For $\eta = 0$, the equation (1) boils down to $\mathcal{W} = \mathcal{W}^\phi$, meaning that we maximize the IP path diversity only. Naturally, this results in the smallest value of the physical vulnerability \mathcal{W}^ϕ . However, the overlay path diversity is not optimized: the overlay vulnerability \mathcal{W}^λ is significantly larger for $\eta = 0$ than for $\eta > 0$. Analogously, for $\eta = 1$ we have $\mathcal{W} = \mathcal{W}^\lambda$, which results in minimal \mathcal{W}^λ , but clearly suboptimal \mathcal{W}^ϕ .

In contrast, for a wide range $0 < \eta < 1$, we observe a plateau of minimal values of both \mathcal{W}^λ and \mathcal{W}^ϕ . This means that the algorithm usually finds a topology \mathbf{G}^λ that minimizes both \mathcal{W}^λ and \mathcal{W}^ϕ at the same time, and thus suits any choice of η .

This is important, because in practice, we always have some non-zero probability of a node failure $p^\lambda > 0$ or IP loss $p^\phi > 0$, which, combined with (3), places us in the regime $0 < \eta < 1$. Moreover, the exact value of η (which might be difficult to estimate in practice) does not affect the resulting topology \mathbf{G}^λ . For these reasons, *in the remainder of this paper we present the results obtained for $\eta = 0.5$, which represents well the entire range $0 < \eta < 1$.*

D. Detailed results for $0 < \eta < 1$

We present two types of results. First, in Fig. 5, under each of the four protection techniques we compare the vulnerabilities yielded by the heuristic, lower-bounds, and *the optimal topology* \mathbf{G}^λ . We find the optimal topology by running an exhaustive search with some obvious and more sophisticated pruning, which allowed us to fully study the overlays of up to 6 nodes in a reasonable time. Next, in Fig. 6 we present the heuristic results obtained for larger systems. We draw the following conclusions C2-C6.

C2: 1-LAYER outperforms RAND.

Recall that 1-LAYER directly minimizes the overlay vulnerability \mathcal{W}^λ , whereas RAND guarantees the feasibility of \mathbf{G}^λ only. Therefore, it is not surprising that 1-LAYER results in values of \mathcal{W}^λ significantly smaller than RAND. In particular, under *MultiTree* we achieve $\mathcal{W}^\lambda \simeq 0$. Interestingly, the physical vulnerability \mathcal{W}^ϕ also benefits under 1-LAYER, resulting in a decrease of 10%-50% (depending on the protection technique).

C3: 2-LAYER outperforms 1-LAYER.

More importantly, when moving from 1-LAYER to 2-LAYER we observe a further substantial decrease in \mathcal{W}^ϕ , ranging from 30% to 70%. This is especially well pronounced in larger systems (see Fig. 6). In contrast, \mathcal{W}^λ remains practically unchanged. This is not surprising, because minimizing \mathcal{W}^λ is exactly the objective of 1-LAYER.

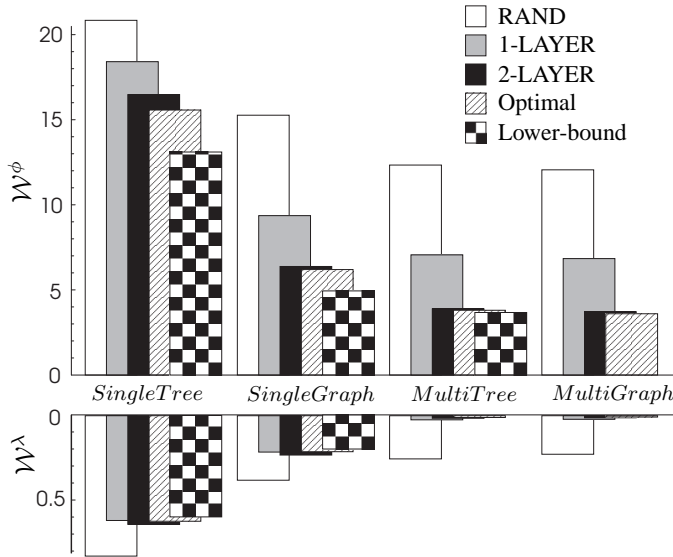


Fig. 5. Vulnerability \mathcal{W}^ϕ (top) and \mathcal{W}^λ (bottom) under each of the four protection techniques, in a small system with $0 < \eta < 1$. We compare the optimal topology (‘Optimal’) with heuristic-based solutions (RAND, 1- and 2-LAYER) and the lower-bounds. We randomly choose $|V^\lambda \cup \{s\}| = 6$ nodes in the DIMES data set, and set $\gamma = 0.5$, $b_{max} = 2$. We use FEC(3,2) in *MultiTree* and FEC(4,3) in *MultiGraph*. The results are averaged over 200 different sets of nodes. The pie-chart shows how often a given protection technique results in the smallest \mathcal{W}^ϕ (out of all techniques). We discard here the cases where more than one technique leads to minimal \mathcal{W}^ϕ .

C4: *MultiTree* is in general the best protection technique.

Consider first the overlay vulnerability \mathcal{W}^λ . We observe that *MultiTree* and *MultiGraph* usually lead to $\mathcal{W}^\lambda = 0$ (with a slight advantage of *MultiTree*). Indeed, to achieve this, it is enough to guarantee that every destination serves as a relay in at most $n - k$ distribution graphs.

Moreover, according to the pie-chart in Fig. 5, *MultiTree* usually achieves the smallest physical vulnerability \mathcal{W}^ϕ out of all three protection techniques. To conclude, 2-LAYER *MultiTree* is a good and practical choice for protecting an interactive ALM.

C5: Nearly optimal results can be achieved already by a simple heuristic.

Recall that in Fig. 5 we show (among others) the optimal results. Interestingly, for all protection techniques the results obtained by our heuristic under 2-LAYER are nearly optimal. Similarly, in Fig. 6, most 2-LAYER results closely approach the corresponding lower-bounds. This means that we do not need any sophisticated and dedicated techniques to find a very good topology \mathcal{G}^λ . On the contrary, a simple general-purpose heuristic (simulated annealing in our case) is sufficient.

C6: The IP-path diversity is limited by the system size $|V^\lambda|$ and redundancy γ_{max} .

It is relatively easy to achieve a very high overlay-level path diversity. In particular, *MultiTree* usually reaches a zero logical vulnerability $\mathcal{W}^\lambda = 0$ (see the discussion in C4).

In contrast, there exist a number of factors that prevent us from achieving a high IP-level path diversity, i.e., that lower-bound the physical vulnerability \mathcal{W}^ϕ . First, \mathcal{W}^ϕ decreases with increasing system size $|V^\lambda|$. This is clearly visible in Fig. 7. We can also observe this tendency in more constrained settings

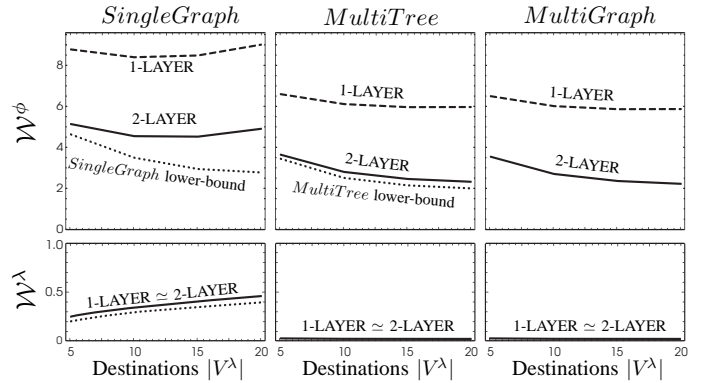


Fig. 6. Vulnerability \mathcal{W}^ϕ and \mathcal{W}^λ in different settings for varying number of destination nodes $|V^\lambda| = 5 \dots 20$. We fix $\gamma_{max} = 0.5$, $b_{max} = 2$ and the regime $0 < \eta < 1$. We use three protection techniques: *SingleGraph*, *MultiTree* with FEC(3, 2), and *MultiGraph* with FEC(4, 3). For clarity, we do not show the *SingleTree* results, as they do not fit in the current scale. For the same reasons we do not show the RAND results for any of the techniques. The results are averaged over 1000 different sets of $|V^\lambda \cup \{s\}|$ nodes chosen at random from the DIMES data set.

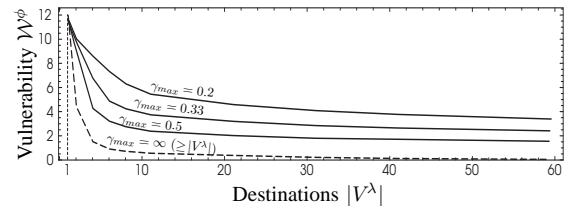


Fig. 7. Minimal Vulnerability \mathcal{W}^ϕ as a function of system size $|V^\lambda|$, assuming $b_{max} = \infty (\geq |V^\lambda|)$ and $t_{max} = \infty$. The dashed curve is the *CompleteGraph* lower bound, i.e., a fully connected *SingleGraph* where all destinations actively forward the source packets to all other destinations. The remaining (plain) curves are *MultiTree* lower bounds for three different values of γ_{max} . The results are averaged over 1000 sets of overlay nodes.

- e.g., in Fig. 6 the curves decline with growing $|V^\lambda|$ (except *SingleGraph*). This phenomenon can be easily explained. With one source and one destination ($|V^\lambda| = 1$) we have no choice but to send the traffic directly to this destination, resulting in virtually no path diversity. But with growing $|V^\lambda|$, we obtain more and more means to exploit the underlying IP path diversity.

Second, the effective IP-level path diversity is limited by the maximal redundancy γ_{max} . In Fig. 7 we show *MultiTree* lower-bounds for three different values of γ_{max} . For example, for $\gamma = 0.2$ we have $\mathcal{W}^\phi \simeq 4$ even for large $|V^\lambda|$, which is much more than roughly $\mathcal{W}^\phi \simeq 0$ for the same setting with $\gamma = \infty$.

VIII. CONCLUSION AND FUTURE WORK

Given the stringent delay requirements and capacity constraints of the interactive ALM, it is (usually) still possible to create a system with a high path diversity at *both* the overlay and the IP layer. This results in a good protection against overlay node failures *and* IP losses. However, such a system should be carefully designed. It is not sufficient to maximize the overlay path diversity, as it does not automatically optimize the IP layer. Therefore, we should take into account not only the overlay-level information, but also the actual mapping of the overlay links on the IP layer. As a result, for real Internet

topologies we reduce the system's physical Vulnerability \mathcal{W}^ϕ by typically 30%-70%, while keeping the logical Vulnerability \mathcal{W}^λ unchanged. Moreover, with the help of a set of lower-bounds, we showed that our results are nearly optimal. In a more general context, we have demonstrated that system size and maximal allowed redundancy are two important factors that naturally limit the available IP-path diversity. Last but not least, we have introduced a novel topology-based metric of Vulnerability \mathcal{W} , that not only captures the system's path diversity in a simple and meaningful way, but also can be formally linked with the average packet loss rate observed at the destinations.

In our future work we plan to consider the all-to-all multicast scenario instead of the single-source one. It would also be interesting to study if the protection techniques that we consider always enable us to find the best possible solution. Maybe there exist other techniques that perform better in some cases? Delay-constrained network coding could be an interesting direction to try.

REFERENCES

- [1] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," *ICMCS '97: Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)*, 1997.
- [2] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *Sigmetrics*, 2000.
- [3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *Sigcomm*, 2002.
- [4] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," *Infocom*, 2002.
- [5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," *SOSP'03*, 2003.
- [6] D. Tran, K. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," *Proc. of Infocom*, 2003.
- [7] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," *IEEE Infocom*, 2003.
- [8] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *ACM NOSSDAV*, 2002.
- [9] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, 2003.
- [10] M. Hefeeda, A. Habib, B. Boyan, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," *ACM Multimedia Conference MM'03*, 2003.
- [11] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," *In Proceedings of IEEE INFOCOM*, 2005.
- [12] R. Zimmermann and L.S.Liu, "Active: Adaptive low-latency peer-to-peer streaming," *SPIE/ACM Multimedia Computing and Networking*, 2005.
- [13] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," *Proc. of Infocom*, 2006.
- [14] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," *Proc. of Infocom*, 2007.
- [15] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007.
- [16] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," *Sigcomm*, 2001.
- [17] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," *Proc. of Infocom*, 2007.
- [18] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," *IEEE Infocom*, 2004.
- [19] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in peer-to-peer gaming," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 79-82, 2007.
- [20] S.-W. Tan, A.G.Waters, and J. Crawford, "Meshtree: A Delay optimised Overlay Multicast Tree Building Protocol," *ICPADS*, 2005.
- [21] S.-W. Tan, G. Waters, and J. Crawford, "A performance comparison of self-organising application layer multicast overlay construction techniques," *Computer Communications*, vol. 29, 2006.
- [22] R. Zimmermann, B. Seo, L. S. Liu, R. S. Hampole, and B. Nash, "Audiopeer: A collaborative distributed audio chat system," *Distributed Multimedia Systems (DMS 2004)*, 2004.
- [23] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive fec-based error control for internet telephony," *IEEE Infocom*, 1999.
- [24] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *In SIGMETRICS*, 2003.
- [25] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," *Proc. Visual Communication and Image Processing, VCIP*, 2001.
- [26] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (pdf) system for packet switched networks," *Proc. of Infocom*, 2003.
- [27] Y. Li, Y. Zhang, L. Qiu, and S. Lam, "Smartunnel: Achieving reliability in the internet," *Proc. of Infocom'07*, 2007.
- [28] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," *Proc. of Infocom'05*, 2005.
- [29] W. Cui, I. Stoica, and R. Katz, "Backup path allocation based on a correlated link failure probability model in overlay networks," *ICNP*, 2002.
- [30] C. Tang and P. McKinley, "Improving multipath reliability in topology-aware overlay networks," *Fourth International Workshop on Assurance in Distributed Systems and Networks (ADSN) (ICDCSW'05)*, 2005.
- [31] T. Fei, S. Tao, L. Gao, and R. Guérin, "How to select a good alternate path in large peer-to-peer systems?" *IEEE Infocom*, 2006.
- [32] Z. Li and P. Mohapatra, "The impact of topology on overlay routing," *Proc. of Infocom'04*, 2004.
- [33] H. Zhang, J. Kurose, and D. Towsley, "Can an overlay compensate for a careless underlay?" *In Proceedings of IEEE Infocom*, 2006.
- [34] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," *In Proc. of ACM SIGCOMM*, 2004.
- [35] Y. Zhang, N. Duffield, V.Paxson, and S. Shenker, "On the constancy of internet path properties," *ACM SIGCOMM Internet Measurement Workshop*, 2001.
- [36] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "An overlay architecture for high quality voip streams," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1250 - 1262, 2006.
- [37] W. Wang, D. A. Helder, S. Jamin, and L. Zhang, "Overlay optimizations for end-host multicast," *Workshop on Networked Group Communications (NGC)*, 2002.
- [38] A. El-Sayed and V. Roca, "On robustness in application-level multicast: the case of hbm," *IEEE Symposium on Computers and Communications*, 2004.
- [39] J. Silber, S. Sahu, J. P. Singh, and Z. Liu, "Augmenting overlay trees for failure resiliency," *Proc. of Globecom*, 2004.
- [40] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu, "Robust and efficient path diversity in application-layer multicast for video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, 2005.
- [41] X. Hoang and Y. Lee, "Dual parent multicast graph for failure resilient peer-to-peer multimedia streaming," *CCNC*, 2006.
- [42] S. Y. Shi, J. S. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," *NOSSDAV '01*, 2001.
- [43] "Dimes," <http://www.netdimes.org>.
- [44] "Planetlab," <http://www.planet-lab.org/>.

APPENDIX A: PROOFS

Proof of Theorem 1 [Equivalence of P1 and P2]:

Denote by $\mathbb{P}(Y)$ the probability that all components (IP links and overlay nodes) in a set $Y \subseteq V^\lambda \cup E^\phi$ fail, and no other component fails. Recall that each overlay node fails independently with probability p^λ and each IP link fails independently

with probability p^ϕ . Thus the probability of no failure is

$$\mathbb{P}(Y=\emptyset) = (1-p^\lambda)^{|V^\lambda|}(1-p^\phi)^{|E^\phi|} = (1-p^\lambda)(1-p^\phi)\beta,$$

where $\beta = (1-p^\lambda)^{|V^\lambda|-1}(1-p^\phi)^{|E^\phi|-1}$. The probabilities of a single failure of an overlay node v' and of an IP link e^ϕ are

$$\mathbb{P}(\{v'\}) = p^\lambda(1-p^\phi)\beta \quad \text{and} \quad \mathbb{P}(\{e^\phi\}) = p^\lambda(1-p^\lambda)\beta.$$

Hence the probability $\mathbb{P}(|Y|=1)$ of exactly one failure is

$$\mathbb{P}(|Y|=1) = \beta \left(|V^\lambda| p^\lambda (1-p^\phi) + |E^\phi| p^\phi (1-p^\lambda) \right).$$

Let $F_v(Y)$ be the indicator function returning 1 when a source packet is *not* delivered at node v within $t_{max}(v)$ due to the failure of Y , and 0 otherwise. For instance, if exactly one IP link e^ϕ fails then $F_v(\{e^\phi\}) = 1_{\{e^\phi \in C^\phi(v)\}}$. Now we can write

$$\begin{aligned} r(v) &= \mathbb{P}(\text{packet is not delivered at } v \text{ within } t_{max}(v)) = \\ &= \sum_{\text{all } Y \subseteq V^\lambda \cup E^\phi} F_v(Y) \mathbb{P}(Y) = \sum_{\{Y: |Y|=1\}} F_v(Y) \mathbb{P}(Y) + R_v = \\ &= \sum_{v' \in V^\lambda} F_v(\{v'\}) \mathbb{P}(\{v'\}) + \sum_{e^\phi \in E^\phi} F_v(\{e^\phi\}) \mathbb{P}(\{e^\phi\}) + R_v = \\ &= p^\lambda (1-p^\phi) \beta \sum_{v' \in V^\lambda} F_v(\{v'\}) + p^\phi (1-p^\lambda) \beta \sum_{e^\phi \in E^\phi} F_v(\{e^\phi\}) + R_v = \\ &= p^\lambda (1-p^\phi) \beta \cdot |C^\lambda(v)| + p^\phi (1-p^\lambda) \beta \cdot |C^\phi(v)| + R_v = \\ &= \alpha \left(\eta \cdot |C^\lambda(v)| + (1-\eta) \cdot |C^\phi(v)| \right) + R_v, \end{aligned} \quad (6)$$

where $\alpha = (p^\lambda + p^\phi - 2p^\lambda p^\phi) \beta$ and $\eta = \frac{p^\lambda(1-p^\phi)}{p^\lambda + p^\phi - 2p^\lambda p^\phi}$.

The remainder R_v in (6) can be upper-bounded as follows:

$$\begin{aligned} R_v &= \sum_{\{Y: |Y|>1\}} F_v(Y) \mathbb{P}(Y) \leq \sum_{\{Y: |Y|>1\}} \mathbb{P}(Y) = 1 - \mathbb{P}(|Y| \leq 1) = \\ &= 1 - \beta \left((1-p^\lambda)(1-p^\phi) + |V^\lambda| p^\lambda (1-p^\phi) + |E^\phi| p^\phi (1-p^\lambda) \right) = \\ &= 1 - \beta \left(1 + p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| + |E^\phi| - 1) \right) \leq \\ &\leq 1 - \beta \left(1 + p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| - 1) (|E^\phi| - 1) \right) = \\ &= 1 - \beta(1+Z) \leq 1 - (1-Z)(1+Z) = Z^2 = O\left((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2 \right), \end{aligned} \quad (7)$$

where

$$Z = p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| - 1) (|E^\phi| - 1),$$

and where we used $(1-p)^x \geq 1 - px$ to lower-bound β by

$$\beta \geq (1-p^\lambda (|V^\lambda| - 1)) (1-p^\phi (|E^\phi| - 1)) = 1 - Z.$$

Finally, combining (6) and (7) we obtain

$$\begin{aligned} \bar{r} &= \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} r(v) = \\ &= \frac{\alpha}{|V^\lambda|} \left(\eta \sum_{v \in V^\lambda} |C^\lambda(v)| + (1-\eta) \sum_{v \in V^\lambda} |C^\phi(v)| \right) + \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} R_v = \\ &= \alpha \cdot \mathcal{W} + O\left((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2 \right). \end{aligned} \quad \blacksquare$$

Proof of Observation 1 [Each protection technique may be best]:

First note that $\gamma = 0.5$ and $|V^\lambda| = 4$ allow us to use (i) *SingleGraph* with $|E_1^\lambda| = 6$ edges (i.e., 2 redundancy

edges), (ii) *MultiTree* with FEC(3,2) or (iii) *MultiGraph* with FEC(4,3) and 2 redundancy edges. Values of n higher than 4 do not make sense because $|V^\lambda| = 4$.

Consider now the setting presented in Fig. 8. We will show that the three possible underlying IP topologies in (a), (b) and (c), the unique optimal protection techniques are *MultiTree*, *MultiGraph* and *SingleGraph*, respectively

For the IP topology as shown in Fig. 8a, *MultiTree* with FEC(3,2) and topology \mathbf{G}^λ as shown achieves $\mathcal{W}^\phi = 0$. In contrast, *SingleGraph* with two redundancy edges must leave two destinations unprotected (with in-degree equal to 1) resulting in $\mathcal{W}^\phi > 0$. Finally, under *MultiGraph* with FEC(4,3) every overlay node contributes to \mathcal{W}^ϕ with $\mathcal{W}^\phi(v) > 0$. With only two redundancy links at our disposal we must leave at least two destinations with $\mathcal{W}^\phi(v) > 0$, resulting in $\mathcal{W}^\phi > 0$

Let us now move to Fig. 8b. For this IP topology *MultiGraph* with FEC(4,3) achieves $\mathcal{W}^\phi = 0$ with the presented topology (note two redundant edges in two of the trees). As before, *SingleGraph* is limited to $\mathcal{W}^\phi > 0$. And clearly, under *MultiTree* with FEC(3,2) we always have $\mathcal{W}^\phi(v) > 0$ and thus $\mathcal{W}^\phi > 0$.

Finally, in Fig. 8c, *SingleGraph* achieves $\mathcal{W}^\phi = 2$ with the presented \mathbf{G}^λ . In contrast, for every destination under *MultiTree* with FEC(3,2) we have $\mathcal{W}^\phi(v) = 2$ and thus $\mathcal{W}^\phi = 8$. *MultiGraph* with FEC(4,3) every destination v must contribute at least $\mathcal{W}^\phi(v) \geq 1$, unless reinforced with 2 or more redundancy edges. As we have only 2 redundancy edges at our disposal, *MultiGraph* results in $\mathcal{W}^\phi \geq 3$. ■

APPENDIX B: LOWER BOUNDS

In this section we derive a number of lower-bounds on Vulnerability \mathcal{W}^ϕ .

A. Simple lower-bound on \mathcal{W}^ϕ

Denote by $E^{in}(v) \subseteq E^\lambda$ (resp., and $E^{out}(v) \subseteq E^\lambda$) the sets of all edges incoming to (resp., outgoing from) node $v \in V^\lambda \cup \{s\}$ in the complete graph G^λ . Our basic component in the construction of a lower-bound is the set $E_{acc}^\phi(v) \subseteq E^\phi$ of *effective access links* of node v , defined as

$$E_{acc}^\phi(v) = \begin{cases} \bigcap_{e^\lambda \in E^{out}(v)} M(e^\lambda) & \text{for } v = s \\ \bigcap_{e^\lambda \in E^{in}(v)} M(e^\lambda) & \text{for } v \in V^\lambda \end{cases}$$

In other words, $E_{acc}^\phi(v)$ is the set of IP links that are traversed by *every* IP path incoming to v (or outgoing from s if $v = s$). The sets $E_{acc}^\phi(v)$ depend on the choice of $V^\lambda \cup \{s\}$.

As any path from s to $v \in V^\lambda$ must always traverse all the effective access links of s and v , the minimal contribution $\mathcal{W}^\phi(v)$ to vulnerability \mathcal{W}^ϕ of each destination node v is

$$\mathcal{W}^\phi(v) \geq |E_{acc}^\phi(s)| + |E_{acc}^\phi(v)|, \quad (8)$$

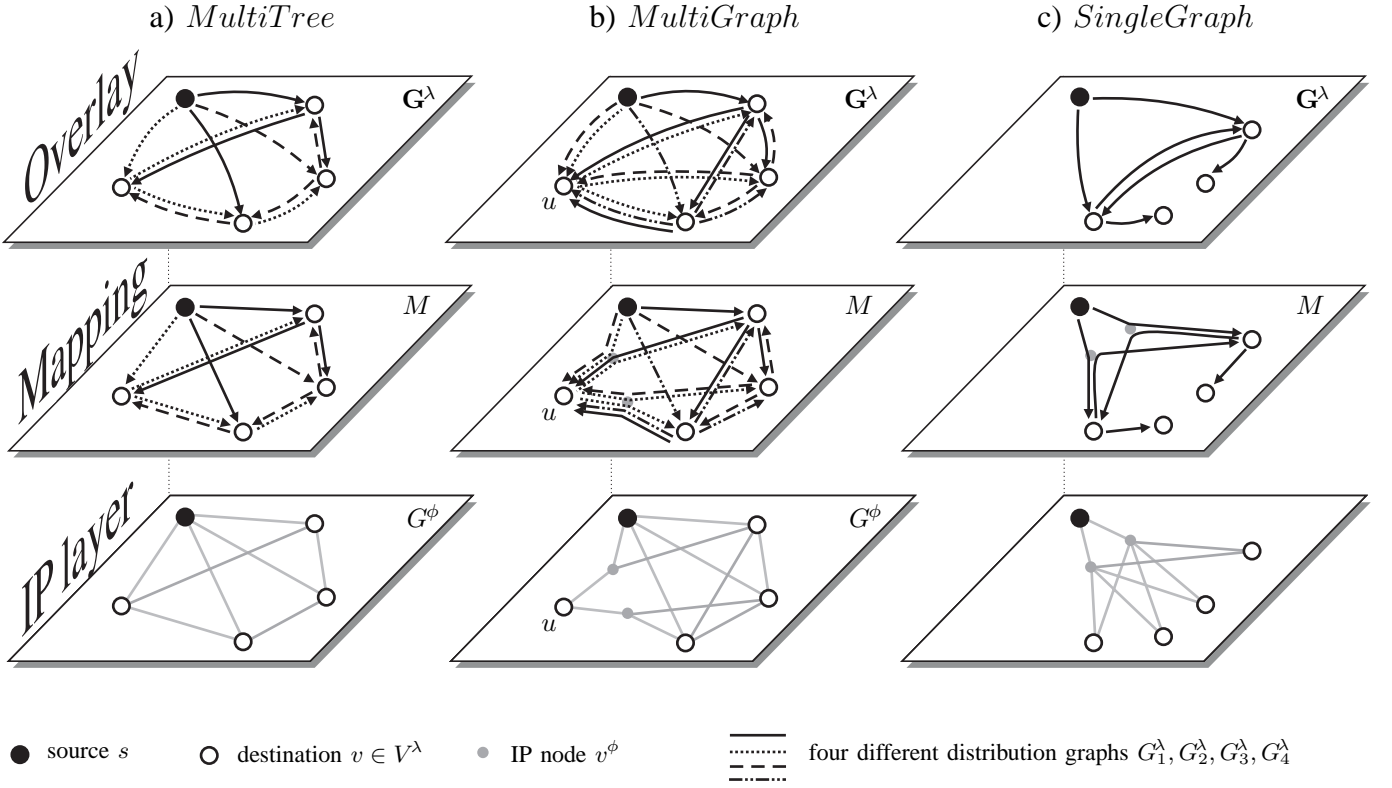


Fig. 8. The absolute and relative effectiveness of the *SingleGraph*, *MultiTree* and *MultiGraph* strongly depends on the underlying IP topology. In this toy example we have one source s and $|V^\lambda| = 4$ destinations. For each of the three possible underlying IP topologies we find the optimal protection technique and topology G^λ , assuming $\gamma = 0.5$, no local capacity constraints and $t_{max} = \infty$. These techniques are *MultiTree*, *MultiGraph* and *SingleGraph* for (a), (b) and (c), respectively.

which gives us the following simple lower-bound on Vulnerability \mathcal{W}^ϕ :

$$\mathcal{W}^\phi \geq |V^\lambda| \cdot |E_{acc}^\phi(s)| + \sum_{v \in V^\lambda} |E_{acc}^\phi(v)|. \quad (9)$$

B. CompleteGraph lower-bound on \mathcal{W}^ϕ

Another approach is to relax all capacity constraints, set $t_{max} = \infty$, and compute Vulnerability \mathcal{W}^ϕ under *SingleGraph* with $G_1^\lambda = G^\lambda$, i.e., the distribution (complete) graph that contains all the possible edges. Clearly, this globally lower-bounds \mathcal{W}^ϕ ; we refer to this as *CompleteGraph* lower-bound.

C. SingleGraph (and SingleTree) lower-bound on \mathcal{W}^ϕ

The bounds given above are general, but usually poor. In contrast, it is possible to derive bounds that are better, but restricted to the chosen protection technique. For *SingleGraph* we can prove the following:

Theorem 2 (*SingleGraph* lower-bound for \mathcal{W}^ϕ): For *SingleGraph* protection technique we have

$$\begin{aligned} \mathcal{W}^\phi \geq & |V^\lambda| \cdot |E_{acc}^\phi(s)| + \sum_{v \in V_a^\lambda} |E_{acc}^\phi(v)| + \\ & + \sum_{v \in V^\lambda \setminus V_a^\lambda} \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)|, \quad (10) \end{aligned}$$

where $V_a^\lambda \subseteq V^\lambda$ is the set of $a = \lfloor \gamma \cdot |V^\lambda| \rfloor$ destinations with the smallest gains $g(v) = \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)| - |E_{acc}^\phi(v)|$.

Proof of Theorem 2 [*SingleGraph* lower-bound for \mathcal{W}^ϕ]: Let $\mathbf{G}^\lambda = \{G_1^\lambda\}$, $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$, $|E_1^\lambda| > |V^\lambda|$ be a system that uses the *SingleGraph* technique. Consider the minimal contribution $\mathcal{W}^\phi(v)$ of each destination node $v \in V^\lambda$ when the protection is provided by allowing for additional links. It depends on the in-degree $indeg_1(v)$ of v in G_1^λ . If $indeg_1(v) \geq 2$ then we reuse the bound (8). In contrast, if $indeg_1(v) = 1$ then we know that any path from s to v must always traverse all the effective access links of s and all IP links in the mapping of its incoming overlay edge e^λ . We can thus lower-bound $\mathcal{W}^\phi(v)$ by taking the edge e^λ with the shortest mapping $M(e^\lambda)$, i.e., if $indeg_1(v) = 1$ then

$$\mathcal{W}^\phi(v) \geq |E_{acc}^\phi(s)| + \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)|. \quad (11)$$

(Note that we exclude $E_{acc}^\phi(s)$ from $M(e^\lambda)$ to avoid a potential double counting of the effective access links of s .) The bound (11) is equal or higher than that in (8). Therefore, in order to find the minimal \mathcal{W}^ϕ we must consider the maximal possible number a of nodes with $indeg_1(v) > 1$. By the global constraint, there are at most $\lfloor \gamma \cdot |V^\lambda| \rfloor$ redundancy edges in the system. As each of them increases the in-degree of one destination by 1, the maximal number of destinations with $indeg_1(v) > 1$ is $a = \lfloor \gamma \cdot |V^\lambda| \rfloor$.

Which a destination nodes should we assign with $\text{indeg}_1(v) > 1$? Again, as we minimize \mathcal{W}^ϕ , we must match $\text{indeg}_1(v) > 1$ with a set $V_a^\lambda \subseteq V^\lambda$ of a destinations with the smallest gains

$$g(v) = \min_{e^\lambda \in E^{\text{in}}(v)} |M(e^\lambda) \setminus E_{\text{acc}}^\phi(s)| - |E_{\text{acc}}^\phi(v)|$$

due to the existence of more than one incoming edge.

To conclude, for the nodes in V_a^λ we use the bound (8) and for all other destinations we use (11); this results in the final formula (10) for the lower-bound on \mathcal{W}^ϕ given in Theorem 2 that we were to prove. ■

D. MultiTree lower-bound on \mathcal{W}^ϕ

Consider now *MultiTree*. This scenario has also some specific features that can be exploited when lower-bounding \mathcal{W}^ϕ . For example, in Fig. 3a (bottom), the set of effective access links of destination u is empty $E_{\text{acc}}^\phi(u) = \emptyset$. However, under *MultiTree* with FEC(3,2), for any topology \mathbf{G}^λ either e^ϕ or f^ϕ must be a critical IP link of u . Therefore in this case $\mathcal{W}^\phi(u) \geq |E_{\text{acc}}^\phi(s)| + 1$, which improves by 1 the bound in (8).

More generally, for each destination node v consider all the IP paths $\{M(e^\lambda) : e^\lambda \in E^{\text{in}}(v)\}$, i.e., mappings of all incoming overlay edges of v . Denote by $q(v, i)$ the total number of different IP links on these paths exactly i hops away from v . By convention, if i exceeds the length $\min_{e^\lambda \in E^{\text{in}}(v)} |M(e^\lambda)|$ of the shortest of the paths then $q(v, i) = \infty$. In Fig. 3 we have $q(u, 1) = |\{e^\phi, f^\phi\}| = 2$ and $q(u, 2) = 3$. For source s the term $q(s, i)$ is defined analogously, except that now we consider $E^{\text{out}}(s)$, i.e., all overlay edges in E^λ outgoing from s . Now we can state the following.

Theorem 3 (MultiTree lower-bound for \mathcal{W}^ϕ): For *MultiTree* protection technique with FEC(n,k) we have

$$\mathcal{W}^\phi \geq \sum_{v \in V^\lambda} \sum_{i=1}^{\lfloor \frac{1}{2} |M(s,v)| \rfloor} 1_{\lfloor n/q(s,i) \rfloor > n-k} + 1_{\lfloor n/q(v,i) \rfloor > n-k}. \quad (12)$$

Proof of Theorem 3 [MultiTree lower-bound for \mathcal{W}^ϕ]:

Let $\mathbf{G}^\lambda = \{G_1^\lambda, \dots, G_n^\lambda\}$, $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda)$, $|E_i^\lambda| = |V^\lambda|$ be a system that uses the *MultiTree* redundancy technique. There are $q(v, i)$ different IP links on all the $|V^\lambda|$ IP paths incoming to v and i hops away from v . At the same time, there are exactly n overlay edges in the system \mathbf{G}^λ that lead to v . Therefore, at least one IP link e^ϕ of the $q(v, i)$ links must be traversed by at least $\lceil \frac{n}{q(v,i)} \rceil$ of these overlay edges. Thus a single failure of e^ϕ results in a loss of at least $\lceil \frac{n}{q(v,i)} \rceil$ FEC packets. So if $\lceil \frac{n}{q(v,i)} \rceil > n - k$ then the original source packet cannot be recovered and this i th hop becomes critical for v and $\mathcal{W}^\phi(v)$ is incremented by 1. Therefore, $\mathcal{W}^\phi(v)$ is not smaller than $\sum_{i=1}^{\infty} 1_{\lfloor n/q(v,i) \rfloor > n-k}$. At the same time all critical elements of the source s are also critical for v , so $\mathcal{W}^\phi(v) \geq \sum_{i=1}^{\infty} 1_{\lfloor n/q(s,i) \rfloor > n-k}$. These two sums can be combined, but we need to guarantee that no critical element is counted twice - on the sides of both v and s . We achieve this by running these sums up to at most half of the length of

the direct path $M(s, v)$, i.e., to $\lfloor \frac{1}{2} |M(s, v)| \rfloor$. This, together with the convention that $q(v, i) = \infty$ for all i larger than the shortest route incoming to v , allows us to write

$$\mathcal{W}^\phi(v) \geq \sum_{i=1}^{\lfloor \frac{1}{2} |M(s,v)| \rfloor} 1_{\lfloor n/q(s,i) \rfloor > n-k} + 1_{\lfloor n/q(v,i) \rfloor > n-k} \quad (13)$$

We obtain (12) by summing (13) for all destination nodes $v \in V^\lambda$, which finishes the prove of Theorem 3. ■

E. SingleGraph (and SingleTree) lower-bound on \mathcal{W}^λ

Let us now consider the lower-bound on logical Vulnerability \mathcal{W}^λ . Interestingly, it is relatively easy to achieve $\mathcal{W}^\lambda = 0$; this is straightforward for *MultiTree* with $t_{\text{max}} = \infty$. This implies that there exist no simple general lower-bound on \mathcal{W}^λ . However, the *SingleGraph* (and *SingleTree*) technique is less effective and usually results in $\mathcal{W}^\lambda > 0$ that can be lower-bounded as follows:

Theorem 4 (SingleGraph lower-bound for \mathcal{W}^λ): For *SingleGraph* we have

$$\mathcal{W}^\lambda \geq \frac{1}{|V|} (|V| - \lfloor \gamma_{\text{max}} |V| \rfloor - \lfloor b_{\text{max}}(s) \rfloor). \quad (14)$$

Proof

There are at least $|V| - \lfloor \gamma_{\text{max}} |V| \rfloor$ destinations with only one incoming overlay link in G_1^λ . For each of these destinations its direct parent is a critical node, unless this parent is the source s itself. As s can have at most $\lfloor b_{\text{max}}(s) \rfloor$ children, there are at least $(|V| - \lfloor \gamma_{\text{max}} |V| \rfloor - \lfloor b_{\text{max}}(s) \rfloor)$ destinations with $|C^\lambda(v)| \geq 1$. This implies (14). ■

Obviously, Theorem 4 holds also for $\gamma_{\text{max}} = 0$, i.e., for the *SingleTree* technique.

APPENDIX C: SPEEDING-UP THE HEURISTIC

In this section we describe two techniques we used to significantly speed-up the heuristic. First, we propose a Bellman-Ford-based algorithm that efficiently finds the critical components and calculates Vulnerability \mathcal{W}^ϕ . Second, we give a useful theorem that allows us to evaluate the effect of a topology change without actually recalculating \mathcal{W}^ϕ .

F. Computation of critical elements for $G_i^\lambda \in \mathbf{G}^\lambda$

A naive approach to find the critical IP links in $G_i^\lambda \subseteq \mathbf{G}^\lambda$ is to consider each IP link $e^\phi \in E^\phi$ (separately), hide all the overlay edges using e^ϕ and run Dijkstra on the resulting graph. This amounts to running Dijkstra $|E^\phi|$ times for every $G_i^\lambda \subseteq \mathbf{G}^\lambda$. Typically $|E^\phi|$ is large and this approach is quite costly, because the heuristic involves numerous evaluations of the objective function (i.e., vulnerability).

Fortunately, there exist better solutions. We propose an algorithm that computes all critical elements in a given distribution graph $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda) \in \mathbf{G}^\lambda$. It is inspired by the Bellman-Ford distance vector routing protocol, but there are three main differences. First, we consider one traffic source s only and many destinations V^λ . Second, every node $v \in V^\lambda$

learns the minimal distance *from* s to v , not from v to s . Third, and most important, our algorithm investigates some additional properties of the discovered paths, i.e., the network elements that these paths avoid. Clearly, if no path from s to v shorter than t_{max} avoids an IP link e^ϕ then e^ϕ is a critical element of v .

1) *Basic version:* Every logical node $v \in V^\lambda \cup \{s\}$ stores a function $t_v : E^\phi \mapsto \mathcal{R}$ defined for all IP links $e^\phi \in E^\phi$. $t_v(e^\phi)$ is the delay on the (so far) shortest path $p_{s,v}^\lambda$ that avoids e^ϕ . If no such path exists (or was found so far) then $t_v(e^\phi) = \infty$. Initially $t_v(e^\phi) = \infty$ for all $e^\phi \in E^\phi$ and $v \in V^\lambda$. In contrast $t_s(e^\phi) = 0$ for every e^ϕ . There are $|V^\lambda|$ iterations. At every iteration, for every overlay edge $e = (v, w) \in E_i^\lambda$ we update update the function t_w for target node w . This means that at j th iteration we find all paths that use j hops. The very basic algorithm is as follows:

```

1 forall  $e^\phi \in E^\phi$  do
2   forall  $v \in V^\lambda$  do
3      $t_v(e^\phi) \leftarrow \infty$ 
4      $t_s(e^\phi) = 0$ 
5   for  $j \leftarrow 1$  to  $|V^\lambda|$  do
6     forall  $e = (v, w) \in E_i^\lambda$  do
7       forall  $e^\phi \notin M(e)$  do
8         if  $t_w(e^\phi) > t_v(e^\phi) + \text{delay}(e)$  then
9            $t_w(e^\phi) \leftarrow t_v(e^\phi) + \text{delay}(e)$ 

```

2) *More efficient implementation:* In practice there are many possible improvements. First, nodes do not have to store full tables. Let initially the domain $DOM(t_v)$ of t_v be empty and change dynamically as follows. Let $t_v(e^\phi) \leftarrow t_{new}$ add e^ϕ to $DOM(t_v)$ (if it was not there yet) and set $t_v(e^\phi)$ to value t_{new} . Let $undefine(t_v(e^\phi))$ delete e^ϕ from $DOM(t_v)$. Finally, let ‘*’ be a default element that covers all elements currently not in $DOM(t_v)$, i.e., $t_v(e^\phi)$ returns $t_v(*)$ for all $e^\phi \notin DOM(t_v)$. Second, in line 6 we may take only those edges whose source nodes were updated at the previous iteration. Moreover, if there are no such edges then we can terminate the algorithm. Third, every value of delay larger than t_{max} can be interpreted as ∞ . With these three improvements the algorithm performs much faster and can be stated as follows:

```

forall  $v \in V^\lambda$  do
   $t_v(*) \leftarrow \infty$ 
   $j \leftarrow 0$ ,  $t_s(*) \leftarrow 0$  {Iteration  $j = 0$ }
  for  $j \leftarrow 1$  to  $|V^\lambda|$  do
     $E'_i \leftarrow \{(v, w) \in E_i^\lambda : v \text{ was updated at iteration } j-1\}$ 
    if  $E'_i = \emptyset$  then return
    forall  $e = (v, w) \in E'_i$  do
      if  $t_w(e^\phi) \leq t_v(e^\phi) + \text{delay}(e)$  forall  $e^\phi \in DOM(t_w)$ 
        then continue {Speed-up}
      forall  $e^\phi \in M(e)$  do
         $t_w(e^\phi) \leftarrow t_w(e^\phi)$ 
      forall  $e^\phi \in DOM(t_v) \cup DOM(t_w) \setminus M(e) \setminus \{*\}$ 
        do  $t_w(e^\phi) \leftarrow \min(t_w(e^\phi), t_v(e^\phi) + \text{delay}(e))$ 
       $t_w(*) \leftarrow \min(t_w(*), t_v(*) + \text{delay}(e))$ 
      forall  $e^\phi \in DOM(t_w)$  do
        if  $t_w(e^\phi) = t_w(*)$  then undefine( $t_w(e^\phi)$ )

```

G. Useful theoretical result

Second, based on the knowledge of the critical element in the current distribution graph G_i^λ , it is often possible to evaluate the effect of a topology change without actually recalculating vulnerability, as follows. Denote by $t_v(e^\phi)$ the delay on the shortest path in G_i^λ from s to v that avoids e^ϕ . If no such path exists then $t_v(e^\phi) = \infty$. (Finding $t_v(e^\phi)$ does not introduce any overhead because it must be implicitly computed during the computation of \mathcal{W}^ϕ .) We can prove that:

Theorem 5 (Effects of edge change on critical elements):

Deleting an overlay link $e^\lambda = (v, w) \in E_i^\lambda$ will not change the configuration of critical elements in G_i^λ if for all $e^\phi \notin M(v, w)$ we have $t_v(e^\phi) + t_{e^\lambda} > t_w(e^\phi)$ or $t_v(e^\phi) = \infty$. And conversely, adding a new overlay link $e^\lambda = (v, w) \notin E_i^\lambda$ may decrease the number of critical elements in G_i^λ only if there exists $e^\phi \notin M(v, w)$ such that $t_v(e^\phi) + t_{e^\lambda} < t_w(e^\phi)$.

Proof of Theorem 5 [Effects of edge change on critical elements]:

First we prove the part of this theorem that speaks about deleting the edge $e = (v, w)$. Assume that for all $e^\phi \notin M(v, w)$ we have $t_v(e^\phi) + t_e > t_w(e^\phi)$ or $t_v(e^\phi) = \infty$. Consider the shortest-delay overlay path $SP(s, w, e^\phi)$ from source s to w that avoids e^ϕ in its mapping. For every $e^\phi \in E^\phi$ we will show that $e \notin SP(s, w, e^\phi)$. We can distinguish three cases:

- 1) $e^\phi \in M(e)$. Then trivially $e \notin SP(s, w, e^\phi)$, because by definition $SP(s, w, e^\phi)$ avoids e^ϕ .
- 2) $e^\phi \notin M(e)$ and $t_v(e^\phi) = \infty$. The latter says that every path from s to v traverses e^ϕ . So every overlay path that contains $e = (v, w)$ must traverse e^ϕ too, yielding $e \notin SP(s, w, e^\phi)$ again.
- 3) $e^\phi \notin M(e)$ and $t_v(e^\phi) + t_e > t_w(e^\phi)$. In this case there exists paths from s to w that contain e and avoid e^ϕ . But, because $t_v(e^\phi) + t_e > t_w(e^\phi)$, none of these paths can be shortest, and thus $e \notin SP(s, w, e^\phi)$.

As for every e^ϕ we have shown that $e \notin SP(s, w, e^\phi)$, the deletion of e will not change the shortest paths from s to w avoiding e^ϕ . So the critical elements of w do not change.

This can be easily extended to every other node $u \in V^\lambda$, because if e belonged to $SP(s, v, e^\phi)$ then just after e the path $SP(s, v, e^\phi)$ would traverse node w . But we know that $e \notin SP(s, w, e^\phi)$, which leads to contradiction.

The second part of this theorem (adding an edge) we can easily prove by contradiction, reusing the arguments above.

■