Graph-based Namespaces and Load Sharing for Efficient Information Dissemination in Disasters

Mohammad Jahanian*, Jiachen Chen†, and K. K. Ramakrishnan*
*University of California, Riverside, CA, USA. Email: mjaha001@ucr.edu, kk@cs.ucr.edu
†WINLAB, Rutgers University, NJ, USA. Email: jiachen@winlab.rutgers.edu

***Abstract*—Timely, flexible and accurate information dissemination can make a life-and-death difference in managing disasters. Complex command structures and information organization make such dissemination challenging. Thus, it is vital to have an architecture with appropriate naming frameworks, adaptable to the changing roles of participants, focused on content rather than network addresses. To address this, we propose POISE, a name-based and recipient-based publish/subscribe architecture for efficient content dissemination in disaster management. POISE proposes an information layer, improving on state-of-the-art Information-Centric Networking (ICN) solutions such as Named Data Networking (NDN) in two major ways: 1) support for complex graph-based namespaces, and 2) automatic name-based load-splitting. To capture the complexity and dynamicity of disaster response command chains and information flows, POISE proposes a graph-based naming framework, leveraged in a dissemination protocol which exploits information layer rendezvous points (RPs) that perform name expansions. For improved robustness and scalability, POISE allows load-sharing via multiple RPs each managing a subset of the namespace graph. However, excessive workload on one RP may turn it into a "hot spot", thus impeding performance and reliability. To eliminate such traffic concentration, we propose an automatic load-splitting mechanism, consisting of a namespace graph partitioning complemented by a seamless, loss-less core migration procedure. Due to the nature of our graph partitioning and its complex objectives, off-the-shelf graph partitioning, *e.g.*, METIS, is inadequate. We propose a hybrid partitioning solution, consisting of an initial and a refinement phase. Our simulation results show that POISE outperforms state-of-the-art solutions, demonstrating its effectiveness in timely delivery and load-sharing.**

I. Introduction

Time and again, it has been shown that properly structured information dissemination can be tremendously beneficial and save lives in disaster management, especially when it comes to timely incident response; an example being the response to the Las Vegas shooting in 2017 [1]. A huge amount of information dissemination during disasters involves distributing it to many participants such as incident commanders, first responders, volunteers and civilians. Thus, having a framework for an efficient dissemination in disasters, that ensures all relevant actors receive the required information in a timely manner is of paramount importance and can be very helpful.

Publish/Subscribe (pub/sub) [2]–[6] systems are a popular means of information dissemination today, and enable one-to-many push-based notification systems. Such a model can be very suitable for disaster scenarios [7], [8]. Support for multicast in the network helps pub/sub with efficiency by greatly reducing network traffic and server/client overhead, compared to server-based poll-based pub/sub solutions [6]. Today, IP multicast is the prevalent multicast protocol, *e.g.*, in IPTV [9], albeit not in the multi-domain case (which may be due to a variety of technical and non-technical reasons). Despite its utility and efficiency, IP multicast has limitations which makes it less than ideal for disaster management pub/-sub: it is tightly intertwined with IP multicast group addresses, does not capture any semantic relationships between groups (*e.g.*, the fact that one group publishes information that is in a subcategory of another group), and is limited by its IP address (sub-)space size. These shortcomings are problematic in disasters where there may be complex command chains with frequent changes and churns in the relationships. This would put excessive burden on publishers and subscribers if IP multicast is used [6]. To overcome these challenges, we use a name-based multicast scheme for pub/sub [6], [7], leveraging the concept of Information-Centric Networking (ICN).

ICN [10]–[12] enables the network layer to understand content names, and provide forwarding and dissemination functionality independent of location, *i.e.*, IP addresses. This location-independent networking is very useful for disaster management [8], [13], [14], since often during disasters, it is the information that matters most to first responders and the victims in need, rather than which point of attachment in the network it originated from. Name-based pub/sub identifies a multicast group by its name, with the namespace capturing the relationship among those names [6]. There are two types of namespace design for name-based pub/sub: topic-based and recipient-based. In topic-based pub/sub (*e.g.*, [6]), a subscriber of a named topic, *e.g.*, "`/CaliforniaWildFires`", is interested in receiving all the content published at a finer granularity, *i.e.*, what is under a particular topic category, *e.g.*, "`/CaliforniaWildFires/WoolseyFire`'. In recipient-based pub/sub [7], a subscriber of a named role, *e.g.*, "`/fireDepartment/fireTeam1`", must receive all messages published to a coarser granularity, *i.e.*, sent to everything above that role in a command chain, *e.g.*, to "`/fireDepartment`". We focus on recipient-based design, as it appropriately models the nature of command chains in disaster response.

Namespace design is an integral part of ICN. State-of-the-art ICN architectures, namely Named Data Networking (NDN) [10], [11] supports strictly hierarchical namespaces, implemented as prefix trees [15]. This hierarchical structure falls short in efficiently modeling complex, multi-dimensional

978-1-7281-2700-2/19/$31.00 2019 © IEEE

namespaces [16], such as today's increasingly complex command chains, where a role (node) can have many dimensions (parents), *e.g.*, time, region, department, *etc.* While it is possible to convert a graph to a strict hierarchy, it will lead to huge amount of redundancy, thus making the management and modification of the namespace extremely costly and inefficient. A study on a Wikipedia dump in [16] shows that converting a typical graph-structured namespace with 10^6 categories (graph nodes) to its hierarchical equivalent, would result in 6.07×10^{54} categories.

We propose POISE, designing graph-based namespaces for in-network name-based pub/sub, with the introduction of an information layer to manage it. While POISE supports both topic-based and recipient-based pub/sub, our focus is on recipient-based pub/sub for disaster management. Additionally, we propose a rendezvous point (RP)-based pub/sub protocol, with the dissemination logic following the graph-based namespaces, to deliver all relevant information to their intended/required recipients (mainly first responders) in a timely manner using push-based multicast. We share the workload among multiple RPs, where each RP is responsible for managing a subset of the namespace graph and functions as the core of the subscription trees associated with those names. Often in disasters, the workload-per-RP distribution is non-uniform and difficult to predict. For example, in a study we conducted on how people used social media (Twitter) during the California Wildfires in 2018, we found out that 70% of tweets asking for help were related to firefighting (rather than police, EMT, *etc.*) which is more than the usual load. In an RP-based pub/sub, this could cause an excessive load on the RP managing names related to firefighting, thus making it a "hot spot" [17]. While it is practically infeasible to optimally balance the load across the whole wide-area network (due to the amount of frequent periodic communication needed), we eliminate such traffic concentration by splitting the congested RP's (*i.e.*, hot spot's) workload: the RP partitions its namespace (sub-)graph with the objective of finding two balanced segments while minimizing inter-RP communication, decides which names to relinquish, and triggers the migration of subscription tree cores related to those names to a new RP or an existing under-loaded RP. Our graph partitioning problem involves calculation of weights for vertices and edges. However, due to the nature of our partitioning formulation, these weights depend on the cut itself; thus making our objective function a "complex" one [18]. As a result, off-the-shelf graph partitioners such as METIS [19], which is regarded as "the gold standard in graph partitioning" [20] and has been widely used for graph-based load splitting and balancing in many application domains [21]–[23], falls short. To overcome this, we propose a hybrid splitting procedure consisting of a heuristic (METIS) and meta-heuristic guided search refinements (using Tabu Search [24]). Our results show the effectiveness of our design.

The key contributions of this paper are the following:

1) A recipient-based pub/sub framework with automatic load splitting for efficient disaster information dissemination.
2) Support for free-form graph-based namespaces and an information layer to capture rich disaster response command chains; our simulation results show that this is more efficient than using state-of-the-art hierarchical namespaces.
3) An automatic name-based load splitting with a novel hybrid workload-driven graph partitioning procedure and a seamless lossless core migration mechanism; our results show the effectiveness and correctness of our core migration, and improved quality and resulting network efficiency of our partitioning procedure, compared to popular off-the-shelf graph partitioning tools.

Disasters potentially cause two major problems for incident management: 1) Network infrastructure can get damaged, leading to intermittent connectivity [14], [25]. 2) Even if infrastructure is intact, the network and servers can experience excessive load and congestion, making many services unavailable [7]. In such situations, traditional means such as 911 operation would be overloaded. Such situations have recently led civilians and victims to use social media (*e.g.*, Twitter) to seek help, having their messages propagated in an un-organized, ad-hoc fashion (*e.g.*, repetitive Re-Tweets) [26], [27]. Our work here primarily addresses the second issue, to have an efficient information organization and load sharing framework that dramatically reduces network load during disasters. Works such as [25], [28] have been proposed to leverage DTN routing with ICN in disasters, to manage infrastructure failures. These works, while orthogonal to ours can be leveraged by POISE. POISE can run on top of such DTN-based protocols in disconnected environments. Additionally, the principles described here may be used in other notification systems with complex name structures as well, such as IoT environments, datacenters, and distributed file systems (such examples described in [29]).

II. Background and Related Work

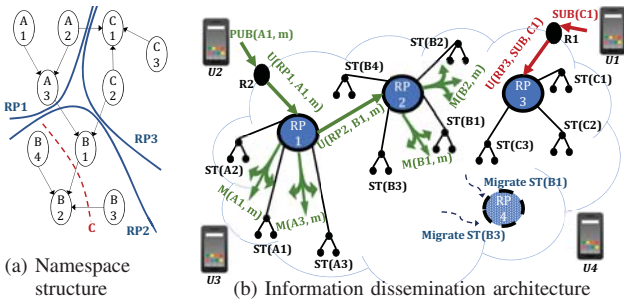
Information-Centric Networking (ICN) [10]–[12] enables access to named objects, independent of their locations. In ICN, contents and entities can be named through identifiers. Having a network layer that recognizes these identifiers can help deliver information without separately establishing an end-to-end communication channel, support in-network content caching, aggregate queries, and provide content-oriented security. All these result in more efficiency, compared to traditional host-centric networks, such as IP. Two notable ICN architectures are NDN [10] and MobilityFirst (MF) [12]. While our proposed information layer can work on top of any ICN or IP architecture, we focus on ICN as it is more efficient for our name-based pub/sub in disaster scenarios [13].

Publish/Subscribe (pub/sub) has become a widely used, popular service over the Internet, in form of RSS feeds, online social networks, *etc.* Most popular pub/sub solutions today are server-based; where subscribers either poll a logically centralized server (via HTTP), or a long-lived connection for timely delivery is maintained [2]. These approaches can be limited in scalability. Broker-based solutions (*e.g.*, ONYX [3], TERA [4]) use an overlay network with distributed brokers, and avoid traffic concentration. However, the dependency

of these solutions on XML data and assertions to decide forwarding paths, couples the information structure with the network layer, making the forwarding function complex. Having pub/sub in the network can help with scalability, and has been proposed for ICNs [5], [6], [30]. ICN with push-based publish/subscribe service models [6] have been proposed. COPSS [6] enhances the query/response model of NDN by allowing consumers to issue a long-standing request, *i.e.*, subscription, for all content related to (subsets of) a name, whenever they are published, and can outperform IP multicast, poll-based methods and flooding-based broadcast [6], [31], in terms of aggregate network load and latency. CNS [7] extends COPSS by introducing recipient-based pub/sub, that can help with information dissemination in disaster scenarios. Our work moves a step further by relieving the strict hierarchy restriction to enable complex free-form graph-based namespaces.

Graph-based information organization has been gaining attention and shown to be important because of its richness and efficiency compared to the more traditional hierarchical structures across multiple application domains. Wikipedia is a very popular and notable example of an information organization system designed as a graph structure: each article can belong to a number of categories, *i.e.*, dimensions [32]. Graph structures for information have been proposed and used in many other contexts as well, *e.g.*, databases [33]–[36], cloud computing [37], and file systems [38]. These works have primarily focused on information organization for storage and indexing. Our work focuses on in-network information organization for name-based information dissemination, extending the current hierarchical structure of NDN [11] to a graph-based one.

Graph partitioning is an important graph operation, and has been an area of research for decades. Optimal graph partitioning is considered to be NP-hard [39]; solutions based on heuristics and approximations exist. The most successful partitioning method is multi-level partitioning [40], [41] (and its tool METIS [19]), which using heuristics, coarsens the graph, does an initial partitioning, and then uncoarsens it. METIS has been widely used for load splitting and balancing in various contexts [21]–[23]. Some use the streaming graph partitioning approach which is used to process partitioning a piece of data on the fly [42], [43]. These algorithms are very fast but their solution quality is lower. This approach is most suitable for extremely large graphs (in the order of trillion vertices). There are approaches using iterative improvement methods for graph partitioning. These methods typically provide high quality solutions. A bad choice of the iterative method and its parameters can make the procedure slow. Work in [44] proposes a graph partitioning algorithm using Tabu search, and shows that it outperforms another popular meta-heuristic method, Simulated Annealing [45], regarding both solution quality and timeliness. Sometimes the objective of partitioning is more than a simple sum of weights, and can be a complex function of the cut itself. This is characterized as the "chicken and egg problem" in [18], as the objective function needed for partitioning decision must be calculated after the partitioning is done; ours belongs to this class. Approaches to



(a) Namespace structure (b) Information dissemination architecture

Fig. 1: A schematic overview of the architecture of POISE.

solve this class of problems have been proposed in works such as [18], [46]–[48] for specific cases. The work in [18] justifies the use of standard partitioning as a good starting point, and then perturb it during the iterative refinement procedures.

Multicast core migration aims at moving a core-based multicast tree [49] from one core (RP) to another, for better load balancing or failure resiliency. Traditional core migration works reported in [50], [51] focus on IP multicast, and primarily focus on low-level migration criteria such as link or node utilization, *etc.* We focus on criteria pertaining to name-level workload in our work. Work in [17] shows the need for RP migration in name-based multicast, but only uses a random load splitting mechanism. We improve it by formulating the workload splitting scheme through a rigorous workload-driven graph partitioning algorithm along with a migration procedure that is reliable and loss-less.

III. Overview of POISE

In this section, we present a brief overview of POISE, as shown schematically in Fig. 1. POISE's namespace supports free-form graph structures, as shown in Fig. 1a, rather than being restricted to the state-of-the-art hierarchical namespaces [11]. This enhancement is possible through decoupling of *information layer* (which manages names and their relations in their natural form, supporting any complex graphs) and the *service layer* (which manages the names used for name-based forwarding at every ICN router), which are coupled together in current Named Data Networks [11]. Each vertex in the graph in Fig. 1a is a name, *i.e.*, a role or attribute, and the edges show relations among them. POISE's information dissemination framework is a name-based pub/sub [7] with the support of name-oriented core-based multicast, with *rendezvous points* (RPs) being the cores for groups; each name also identifies a multicast group. In addition to being the core for the multicast tree (similar to traditional PIM-SM [49], NDN COPSS [6], or MF multicast [30]), POISE's RPs operate at the information layer; in other words, they are information-layer-enhanced RPs. As shown in the example in Fig. 1, the namespace is shared among three RPs (*i.e.*, RP1, RP2, and RP3), each RP managing the sub-graph it is responsible for, and maintaining the subscription trees associated with each of the names, *i.e.*, groups, it is hosting; *e.g.*, RP1 is the core for the subscription tree (ST) for $A1$, $A2$, and $A3$. A *name-to-RP* mapping resolution service resolves a name in the namespace to the RP it is hosting; *e.g.*, the name $C1$ would be mapped to RP3.

The subscription path is shown in Fig. 1b (in red); user $U1$ wants to subscribe to $C1$ (which implicitly means subscribing to all ancestors of $C1$ in the namespace as well). $U1$ sends this request as $SUB(C1)$, without the need to know which is the associated RP or where it resides. $U1$'s first hop router R1 performs the resolution and relays the request as a unicast (U) message to the correct RP, *i.e.*, RP3; thus, $U1$ *joins* $ST(C1)$. The publication path is also shown (in green); $U2$ wants to publish message m to all subscribers of name $A1$ (which implicitly means publishing to subscribers of all descendants of $A1$ as well). $U2$'s first hop router relays this publication as a unicast message to its corresponding RP, namely RP1. Expanding $A1$ to its descendants (*i.e.*, name expansion), it sends m as a multicast (M) downstream to $ST(A1)$ as well as $ST(A2)$. Additionally, RP1 recognizes that there is an edge leading from $A1$ towards a name outside RP1. Thus, RP1 sends a unicast message for this name, $B1$ to its RP, RP2. Note that RP1 only has visibility of namespace up until $B1$ and not further. Performing a similar name expansion, RP2 processes the received request by going through its namespace, which leads to multicasting m downstream along $ST(B1)$ and $ST(B2)$. Thus, users for both subscription and publication scenarios need only send *one packet*, destined to only *one name*; the network takes care of expanding the packet to additional names, if needed. More details on the information layer design and pub/sub dissemination are provided in §IV.

Each RP's workload has a correlation with the part of the namespace it is managing. However, additionally, the load-per-name distribution is likely to be non-uniform, and hard to predict. To address this, another important feature of POISE, *automatic load splitting* is performed to eliminate traffic concentration. Consider the case when RP2 encounters a large amount of workload exceeding its threshold, thus making it a *hot spot*. Triggered by this, RP2 will perform a *partitioning* procedure on its own sub-graph, to provide two balanced *segments*, shown as *cut* C in Fig. 1a. It thus decides to keep $B2$ and $B4$, and relinquish $B1$ and $B3$ to another RP (*e.g.*, RP4), which can be a regular ICN router configured to be a new RP for this environment. As a result, the subscription trees for $B1$ and $B3$ will be migrated to RP4, via a *core migration* procedure. The name-to-RP mapping will then be updated accordingly. More details on the load splitting procedure are in §V.

While security is an important issue while managing disasters, in this paper we mainly focus on the dissemination and load sharing mechanisms. We assume all participants follow the protocols honestly. We will work on enhancing our design with security, as part of our future work.

IV. Namespace and Pub/Sub Design

A. Information Layer and Graph Namespace

POISE supports free-form graph namespaces with their natural structure for in-network information-centric dissemination, without the need to restrict them to any particular data structure, such as a hierarchy or prefix tree as NDN [11], or NDN-based solutions such as CNS [7] do. Fig. 2 is a simple namespace of an incident management command structure for

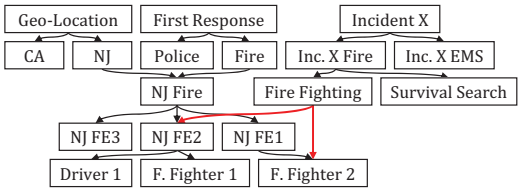


Fig. 2: Graph-based namespace: incident command chain example.

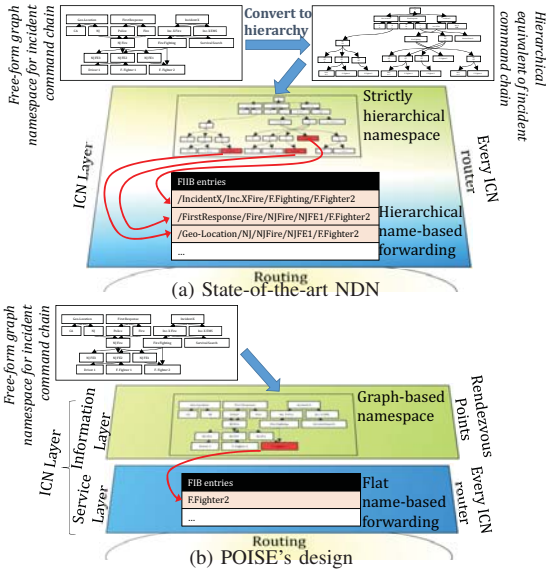


Fig. 3: Design choices for ICN layer.

a disaster management scenario. As we can see, it does not follow a strict hierarchy. The incident management structure is a directed graph, with each node in the graph being a name that denotes a role. The higher levels denote coarser granularity (*e.g.*, "`Fire`" is a broad organizational role for everyone having something to do with fire-related issues), while the lower levels denote finer granularity (*e.g.*, "`NJ FE1`" denotes a specific fire engine dealing with a fire-related task in New Jersey). Edges in the graph represent name relationships and the direction of the edges show the flow of control in the command chain; *e.g.*, "`NJ FE1`" (NJ fire engine 1) is a higher-level authority than "`F.Fighter2`" (fire fighter 2). This representation of the namespace captures the different dimensions in *one* graph, *i.e.*, time, region, department, *etc.* Modification of the namespace is achieved through addition, modification and/or deletion of nodes and/or edges in the graph. In the namespace graph shown in Fig. 2, two sub-namespaces, one organizational, on the left-hand side, and one incident-specific, on the right-hand side, are connected through the two edges shown in red. These two red edges represent the fact that the incident commander has dispatched "`NJ FE2`" and "`F.Fighter2`" roles to take care of Incident X's "`FireFighting`" tasks, shortly after it occurred.

Support for graph namespaces in information dissemination is made possible in POISE through a decoupling in the ICN layer and the introduction of information layer, as shown in Fig. 3. In current name-based network architectures, in particular NDN, the information layer and service layer functionalities are coupled together in the ICN layer, supported

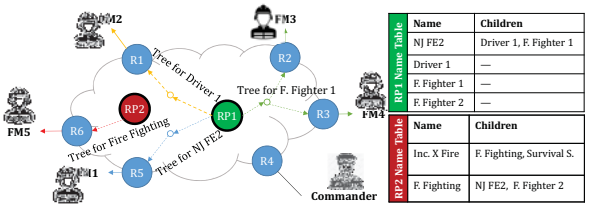


Fig. 4: Example network topology: 5 Firemen subscribe to different roles in the namespace in Fig. 2 and 2 RPs share the workload.

at every ICN router, as shown in Fig. 3a. This restricts the in-network namespace to one particular structure. This means every command structure of the organizations (*e.g.*, namespace in Fig. 2) needs to be converted to a strict hierarchy. As can be seen, this conversion makes the namespace unnecessarily larger and more complex to manage and use; *i.e.*, the "`F.Fighter2`" node in Fig. 2 will appear 3 times in the hierarchical equivalent on each reachable path, and will populate 3 entries in the FIB tables, with a separate entry for each (in this case, 3) different path leading from any root to node "`F.Fighter2`". The FIB table provides the name-based forwarding functionality leveraging the lower layer routing mechanism used for navigating the packet to its relevant locations. In contrast, in POISE (Fig. 3b), we decouple information and service layer functionalities of the ICN layer. Only RPs (designated ICN routers that perform name expansion) need to understand and maintain name relationships in the graph. This design choice in POISE brings a number of great benefits:

1) The ICN-layer namespace in Fig. 3b is more natural, and thus simpler and smaller than its converted equivalent in Fig. 3a. Each name appears once rather than multiple times. This makes managing the namespace more efficient and scalable, as well as its modification less costly, which is important in the highly dynamic disaster scenario.

2) Complicated role relationships, which is prevalent in disaster management, leads to larger FIB tables in Fig. 3a compared to Fig. 3b. As seen in the Fig., "`F.Fighter2`" appears 3 times in Fig. 3a's FIB tables *vs.* once in Fig. 3b's. Thus, POISE consumes less ICN router memory.

3) The above difference also leads to higher number of subscription and publication messages in Fig. 3a, when it comes to recipient-based pub/sub, which we use here. Subscribing (publishing) to role "`F.Fighter2`" would result in 3 subscription (publication) messages in NDN's hierarchical naming framework as seen in Fig. 3a, while it leads to only one message in Fig. 3b. This makes POISE more efficient in a disaster scenario for pub/sub as it reduces the number of messages, thus reducing the network/user load.

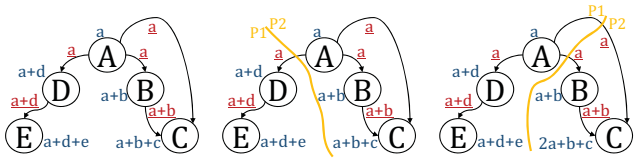
B. Recipient-based Pub/Sub

POISE uses recipient-based pub/sub [7], enhanced with an information layer supporting graph-based namespaces. In name-based pub/sub, subscribing to a name means implicitly also subscribing to a set of names related to that specific name, in accordance with the namespace. POISE's pub/sub logic follows the command chain graph-based namespace. Typically, first responders and volunteers subscribe to (listen to) names, and civilians and incident commanders publish to names. Given the namespace in Fig. 2, subscribing to "`F.Fighter2`", implicitly means also subscribing to all of its ancestors, *i.e.*, "`NJ FE1`", "`FireFighting`", "`NJ Fire`", *etc.* Conversely, publishing to "`NJ Fire`", implicitly means also publishing to all of its descendants, *i.e.*, "`NJ FE1`", "`NJ FE2`", "`F.Fighter2`", *etc.* Expanding a name to all of its descendants on the publication path according to the namespace graph, is performed by the RPs, in a load-shared way. This design is beneficial in disaster management where dynamically-formed interacting groups and individuals involved need to be notified with messages relevant to their tasks in a timely manner, whenever they are published or available, making sure maximum coverage and accuracy is achieved.

To demonstrate the protocol exchange for the information dissemination of POISE, we use a small example: Let us consider the namespace graph in Fig. 2, and the topology in Fig. 4, where we have 5 firemen (FM1-FM5), and one commander, all connected to the network via routers R1-R6. These firemen subscribe to different names: FM1→"`NJ FE2`", FM2→"`Driver1`", FM3 and FM4→"`F.Fighter1`", and FM5→"`FireFighting`". There are two RPs, with each of their name tables shown (partially) in Fig. 4.

The commander wishes to send a message to the name "`FireFighting`", for it to be received by all relevant firemen. When a publication is sent to "`FireFighting`", the message will be sent to the RP serving it, namely RP2. RP2 searches its name table to find out the reachable subgraph visible to it under "`FireFighting`", via BFS/DFS traversal. It sees that it needs to forward the message to "`FireFighting`", "`NJ FE2`" and "`F.Fighter2`". Although the name should be further expanded under "`NJ FE2`", we do not require RP2 to do this. RP2 would forward the message as a multicast to "`FireFighting`" based on the subscription, since "`FireFighting`" is served by itself. It then sends 2 publications to "`NJ FE2`" and "`F.Fighter2`" since it cannot find the entries for these names in the name table. These messages will reach RP1 based on the underlying network performing the lookup (*e.g.*, NDNS lookup in NDN [52] or GNRS lookup in MF [12]) to find that RP1 is responsible for these names, and will then get expanded at and by RP1.

An important difference between graphs and (hierarchical) trees is the possibility of existence of cycles in graphs, which needs to be addressed when dealing with graph-based namespaces. While semantically it is a poor design to have cycles in the namespace graph, it is possible that due to very frequent changes in the namespace, it ends up having cycles (at least during transients), which can lead to loops in the publication dissemination paths. ICN routers, as in NDN, have inherent support for detecting and discarding looped packets with the use of Nonces [15]. POISE uses a similar data plane approach for resiliency against namespace graph cycles: fresh Nonces are used for each new end-user-generated publication and carried in all subsequent expanded packets associated with it. RPs maintain a list of <name, Nonce> pair for (publication) packets they have seen; if an RP encounters a packet with



(a) Before partitioning (b) Partitioning 1 (c) Partitioning 2

Fig. 5: Partitioning impacts multicast workload weight of names.

name and Nonce matching any entry in the list, it will discard it, thus eliminating the possibility of infinite loops.

V. Automatic Load Splitting

A. Partitioning Namespace Graphs

POISE's namespace graph partitioning aims at distributing the load among RPs if traffic concentration overloads an RP. Partitioning is performed locally on the congested RP, only on the (sub-)namespace that it is hosting. We mainly use the monitoring of recent queue size at RPs to measure its load, and use the recent multicast and unicast workloads (explained later in this section) to label the graph for partitioning. Full details of algorithms and mathematical proofs are in [29].

1) Problem Description and Solution Overview: We leverage graph partitioning algorithms to determine which part of the namespace should reside at which RP for load splitting. We treat the namespace as a directed graph with weights on nodes and edges. The initial (input) vertex weights represent the messages sent to each name explicitly from publishers (we call it *incoming unicast load*). To determine the number of messages multicast from a node (called *multicast workload*), we need to consider the incoming unicast load of all of its ancestors. The weight of the edges going out of a name are set to be the multicast workload of that name. The total weight of the edges going out of an RP to other RPs represents the total amount of outgoing inter-RP communication (we call *outgoing unicast load*). We try to balance the sum of multicast workload and outgoing unicast load, in the two partitioned segments and minimize the cut cost. A complexity here is that the decision of the partitioning can alter the weights, *i.e.*, "the chicken and egg problem" [18], thus making the off-the-shelf graph partitioners inadequate; we explain this with an example.

Fig. 5 shows a simple namespace graph at different stages. Let us denote the incoming unicast load of each name (node) as a, b, c, *etc.* Assuming no partitioning (*i.e.*, whole namespace in same RP), the multicast workload of each name is shown in Fig. 5a (blue labels next to each name). *E.g.*, name `C` has to send out publications related to itself, `A`, and `B` to its subscribers; thus making its multicast workload $a+b+c$. Edge weights in Fig. 5, denoting the RP-to-RP communication on that link in case it gets cut, is shown in red and is underlined.

Now let us consider partitioning the graph shown in Fig. 5. There can be many ways to partition a graph. Two examples are depicted in Fig. 5b and Fig. 5c. As seen in the figures, the result of multicast workload of name `C` (and thus the total cost of the resulting graph) differs in the two figures; it is $a + b + c$ in Fig. 5b and $2a + b + c$ in Fig. 5c. The one extra message `C` receives from `A` in Fig. 5c is due to the fact that in this scenario, `B` relays what it has received from `A` to `C`, not knowing that `A` is also a parent of `C`; while in Fig. 5b the graph cut is in a way that it does not cause such duplication. This shows that the graph's multicast workload weight values are a function of partitioning itself; thus, a standard partitioning tool with fixed weights is not sufficient to solve our partitioning problem. We propose the use of a hybrid approach of heuristics (classic graph partitioning) followed by a refining meta-heuristic (guided search).

Our algorithm supports bi-partitioning as well as k-way partitioning (with $k > 2$). Typically, bi-partitioning is preferred over k-way partitioning in POISE, since: 1) it is less computationally complex and more importantly 2) starting from one RP in the beginning, it leads to fewer RPs (which means less inter-RP communication). Additionally, POISE does a *local* partitioning, using only the information at the triggered RP. However, additional information from neighboring RP's can be added and considered for the partitioning decision if needed. An extreme case of that, however, meaning a global partitioning and placement using all the information in the network, while theoretically possible in our graph partitioning approach, is practically not feasible as it requires too much communication to exchange data, which is not desirable in our network environment. Thus, we focus on local, bi-partitioning.

2) Graph Partitioning Procedure: To prepare the graph for partitioning, the RP labels its local namespace sub-graph, which mainly consists of calculating and assigning multicast workload weights to vertices, and unicast workload weights to edges, as the example in Fig. 5 shows. The calculation of weights are done through an iterative diffusion method which follows the propagation logic described in §IV-B. More details on the calculation formulae are provided in [29]. The weights are calculated for each solution instance, including the initial solution provided by METIS [19]. We use Tabu search for iterative refinement of our graph partitioning solutions [24], [44], as described in Algorithm 1. Each solution (candidate) of the procedure provides a cut, which partitions the RP's namespace sub-graph into two *segments* (assuming bi-partitioning).

Algorithm 1 Graph partitioning procedure of POISE

1: Start with an *initial solution* (a partitioning solution),
2: Calculate the *neighborhood* solutions by picking vertices to *move*, *i.e.*, put in the other partition.
3: Calculate the objective function of all neighbors.
4: Filter out the moves in the Tabu list, unless the Tabu move satisfies the *aspiration criteria*, *i.e.*, if it is better that the best solution so far.
5: Pick the neighbor with lowest objective function, as next move candidate.
6: Tabu the picked move for a number (*Tabu tenure*) of iterations.
7: Go to 2 if *stop criteria* has not been met.
8: Report minimal objective and the corresponding partition.

Initial solution: Tabu search typically starts with a random initial solution and improves it. To get a better initial partition [18], we try to use the result from the problem closest to ours – the (static) multi-criteria graph partitioning where the weights will not change according to the partition decisions. We use METIS for this stage as it is a highly popular tool that has been shown to be fast, while providing high quality solutions.

Objective Function: As mentioned, to reduce the search space, we adopt a bi-partitioning approach, where the heavily loaded RP's namespace is partitioned to be split between two RPs, *i.e.*, the current RP and the new RP. The objective (fitness) function we use to evaluate our partitioning solution, takes into account the cost of both segments (sub-namespaces managed by the two RPs) and provides a combined measurement of 'minimizing the imbalance between the two RPs', 'minimizing the maximum single segment load', and 'minimizing the inter-RP communication' (G_1 and G_2 are the two segments, associated with the two RPs):

$$F(G_1, G_2) = \alpha \cdot |TC(G_1) - TC(G_2)| + \beta \cdot max(TC(G_1), TC(G_2)) + \gamma \cdot (UC(G_1) + UC(G_2)) \quad (1)$$

where α, β, and γ are optimization coefficients. Setting higher coefficients for some of the terms would result in the final solution being impacted more by those terms. However, the coefficients can be adjusted. We set all of them to 1 in our test cases, since these values appeared to provide reasonably good benefit, in our experiments. The aim is to minimize F. Function $UC(G_i)$ (segment total unicast cost) is the sum of cut edge weights initiated in G_i, and $MC(G_i)$ (segment total multicast cost) is the sum of all vertex weights in G_i. Furthermore, total load cost of a segment would be:

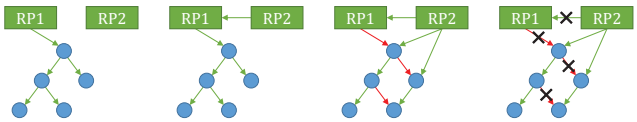
$$TC(G_i) = UC(G_i) + MC(G_i) \quad (2)$$

Stopping criterion: We allow both fixed and adaptive stop criteria. If fixed, a parameter *Max Iterations* i is pre-defined, and Tabu search stops when i is reached. Our adaptive stopping criterion, on the other hand, starts with an *Iteration Base* b, and any time the 'so far found best solution' is changed, b gets added to the current iteration number and makes up the new final iteration number. This ensures that our Tabu search procedure stops only after running with b iterations of no improvement. To prevent the Tabu procedure to keep iterating indefinitely, with this adaptive stopping criterion, an upper bound on the number of iterations is also specified.

B. Migrating Cores

Once the graph partitioning is done, the names in one segment need to be migrated to another core. The RP selection function is similar to that in IP multicast [50], [51]. It may be performed by a network manager or calculated by a Network Coordinate function such as [53]. Once the RP is selected, the process essentially migrates the names in the partitioned subspace to the other RP. However, this has to be done carefully because if a router discards the original subscriptions before it receives all the publications that are in-flight (before the original 'pipe' is drained), these publications will be lost.

To address this issue, we propose a 3-stage (make-before-break) solution to ensure reliable delivery during migration, as shown in Fig. 6. Before migration, we assume there is a multicast tree rooted at RP1 (Fig. 6a). When RP1 decides to move a name to RP2, in stage 1 (Fig. 6b), it notifies RP2 *and* also subscribes to RP2 (creating new green line). Meanwhile, it notifies the network that RP2 is now serving that name (routing update in IP, FIB propagation in NDN, or a GNRS update in MobilityFirst). RP2 now becomes the RP for the name, and routers with the new RP information will send publications to RP2. However, reusing the original multicast tree, we continue to make sure that the publications are delivered during the transient phase. Routers may have not yet updated the name-to-RP mapping, and there can be publications in-flight during the mapping update. Thus, some publications to those names will still reach RP1. We adopt the late-binding concept of MobilityFirst: when an RP receives a publication that is not served by itself. It hands the publication back to the network to then be forwarded to the correct RP accordingly.



(a) Before (b) Init (c) After M1 (d) After M2

Fig. 6: Reliable RP splitting: RP1 relinquishing a name to RP2.

TABLE I: Solution quality of alternatives and global optimum.

Vertices	Edges	METIS	POISE	Optimum	Average	Median
10	14	2,093	1,916	1,916	3,657.21	3,770
20	42	18,905	9,342	9,342	31,888.88	32,055

Then, at stage 2 (the 'make' stage), RP1 sends out a special marker packet (we call it M1) to all the nodes in the subscription tree. M1 is treated just as a normal multicast packet. To make sure that all the subscribers in the tree receive the M1 marker packet, RP1 has to send that packet after it is sure that the new mapping has propagated into the network and the subscriptions based on the old mapping have joined the tree. On receiving M1, routers subscribe towards the new RP and mark the original ones as 'stale' if the original entry in the subscription table is different from the new entry. Fig. 6c shows the subscription after M1 is propagated to the network. The green arrows are the new subscriptions and red arrows are the 'stale' ones. Note that while we mark the subscriptions as stale, we do not delete them. When RP2 sends publications, it sends them along all the subscription links, to ensure delivery. A nonce can be used in the packets to eliminate redundant traffic during this transient phase.

After all the nodes subscribe to the new RP, RP1 can send a second marker packet (we call it M2) to start the third and final stage (the 'break' stage). On receiving this marker packet, the intermediate nodes clean up the 'stale' subscriptions (as is shown in Fig. 6d). When a node has no downstream subscriptions (*e.g.*, RP1 in the Fig.), it will unsubscribe from the upstream naturally. Since all the nodes have subscribed to the new RP, the M2 marker packet acts as the last packet in the pipe. Thus, we will not lose packets if we close the 'pipe' (unsubscribe) after we receive M2.

It is also important to provide resiliency to RP failures. POISE's RP splitting mechanism can be used for recovery. The namespace managed by an RP would have to be backed up and replicated (possibly pro-actively) at a backup router. On detecting an RP failure, the backup router can become active and using the above protocol, the subscription trees for that part of the namespace would be shifted to the backup RP. This would be transparent to publishers and the protocol minimizes loss of in-transit packets.

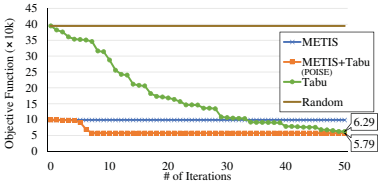


Fig. 7: Partitioning effectiveness.

TABLE II: Quality of METIS *vs.* POISE: Objective function.

Vertices	Edges	METIS	POISE
10	14	2,093	1,916
10	18	2,988	2,319
10	28	5,170	2,873
50	75	11,159	3,820
50	84	99,292	57,897
100	191	25,858	20,470

TABLE III: Average notification latency & aggregate network traffic.

Solution	Average Notification Latency (s)	Aggregate Network Traffic (Gb)
POISE	0.018	492.39
Graph - 1RP	2.741	483.08
Graph - Random Split	4.725	625.69
Hierarchical (CNS)	247.742	866.27

VI. Evaluation

To evaluate POISE, we compare it to a number of existing and theoretical alternatives. In terms of overall architectures, we compare POISE to NDN/CNS [7], a recipient-based push-based pub/sub architecture for notification systems, which is the closest architecture to ours. For namespaces, we compare POISE's graph-based naming with the most advanced state-of-the-art ICN naming, which is NDN's hierarchical naming (as in CNS as well). For load-splitting, we compare POISE to the most popular graph partitioning tool METIS [19]. We use the same design principles of the simulator in [7], while adding the functionality of our information layer graph namespace design and splitting procedures. Our simulator is open-sourced and available in [54]. For the partitioning component, we use the current implementation of METIS [19], plus our refinement and weight/objective calculation procedures.

A. Graph Partitioning Algorithm Evaluation

In this section, we evaluate the quality of POISE's graph partitioning, *i.e.*, the hybrid "METIS+Tabu" algorithm. To compare the quality of solutions provided by METIS, and METIS+Tabu (starting from METIS and then doing a Tabu search) with the global optimum (using exhaustive search), we used two small graphs with 10 and 20 vertices as examples (taken from [55]), as described in Table I. METIS+Tabu uses v iterations with Tabu tenure of $\sqrt{v}$ for a graph with v vertices. For finding the global optimum, we generated all possible solutions using a brute-force (exhaustive search) approach. We also compare with average and median of all possible partitionings, using the metrics of the average (and median) of the objective function (Eq. 1) across all possible solutions. As seen for the two cases in Table I, METIS+Tabu (the approach in POISE) finds the optimal solution. Considering the complexity of the Tabu search and the brute-force approach, we see the significant benefit of using our approach. While the brute-force approach finds the global optimum by checking $2^{n-1}-2$ solutions (assuming bi-partitioning and filtering out of duplicate permutations and no-cut solutions), Tabu finds it by checking $O(iv)$ candidate solutions, which in our case is $O(v^2)$, since we set the number of iterations i to be $O(v)$ and at each iteration, $O(v)$ neighboring solutions are visited and evaluated. The table also shows that the METIS solution is better than the average (or median) solution but is worse than the METIS+Tabu solution (and the global optimum).

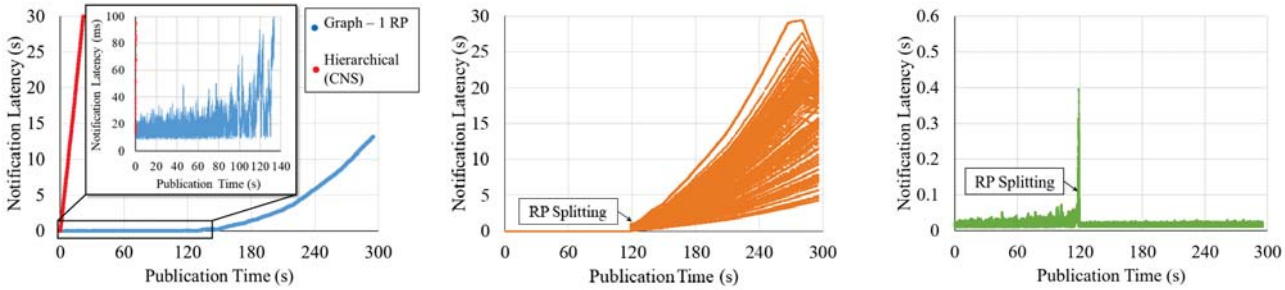
Finding the global optimum through brute-force search is not computationally feasible for large graphs, as the number of candidate solutions to visit grows rapidly exponentially. Even though in Table I, METIS+Tabu found the exact global optimum for the examples examined, this does not necessarily have be to the case for all input graphs. For larger graphs, we only need to do a comparative evaluation, showing that METIS+Tabu finds relatively better, and in most cases, significantly better solutions, than alternative approaches. To show this comparison, we use one of the graphs available online in the repository in [55] (from its "AT&T graphs" package). It is a directed graph with 50 vertices and 84 edges. The graph is unweighted, so we assign random values between 0 and 100 to each vertex, to denote the incoming unicast load for each name. Fig. 7 shows the comparison across different alternatives, in terms of the quality of solution (objective function) for this graph. The following scenarios are used: Random (average of three randomly generated solutions), Tabu (average of three runs of Tabu-only starting from random initial solution), METIS, and METIS+Tabu (POISE). Note that the "Random" and "METIS" scenarios are not iterative procedures therefore achieve a fixed solution quality. We vary the number of iterations on the Tabu-based solutions, with a fixed stop criteria, to show how quickly the search approaches converge to an asymptotically good quality solution.

Fig. 7 shows that METIS+Tabu outperforms the rest. Using METIS as initial solution (METIS+Tabu) *vs.* starting from a random initial point (Tabu) is very effective as the algorithm reaches its asymptotic convergence point (for the range of iterations we examined) much faster (with fewer iterations). The Tabu-only method outperforms METIS only after a relatively large (34) number of iterations. The METIS+Tabu approach outperforms METIS very early, after just 5 iterations. The random partitioning solution is much worse than the other alternatives. Fig. 7 also shows the importance of choosing an appropriate stop criterion. The number of iterations being too small precludes reaching a good solution, and it being excessively large results in waste of time and compute resources.

We have tried our algorithm on a number of different graphs. Table II summarizes results for several of those graphs (taken from [55]) of different sizes. For these cases we use the adaptive stop criterion. For each graph with v vertices, we choose the base iteration number to be v, Tabu tenure to be $\sqrt{v}$ and upper limit on iterations to be $10v$. Comparing the quality of solution shows that our approach (METIS+Tabu) achieves better solutions, especially for the larger graphs.

B. Overall Solution Evaluation

To evaluate the performance of POISE, we implemented an event-driven, packet-level simulator. The simulator supports name-based pub/sub, exploring different alternatives within that paradigm, using any type of multicast network layer underneath. We can compare name-based multicast to



(a) Hierarchical & Graph - 1RP (b) Graph - Random Split (c) POISE

Fig. 8: Notification latency over time in different solutions (Note the difference in the scale of notification latency in POISE).

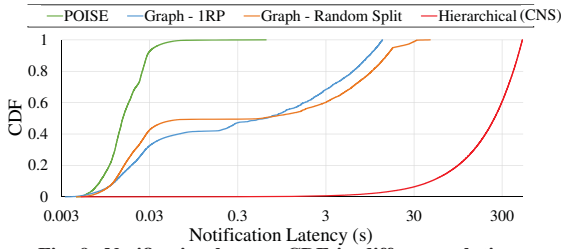


Fig. 9: Notification latency CDF in different solutions.

alternatives such as pull-based pub/sub, IP multicast-based pub/sub, and broadcast-based pub/sub such as those examined in [6]. To evaluate the behavior, we needed a realistic network environment with a number of forwarding routers and end-points that are publishers and subscribers. For this, the network topology we use to evaluate POISE and compare with various alternatives is the Rocketfuel 1221 Telstra [56] with some modification for a state-wide disaster scenario. Our topology contains 46 core routers, with additional 231 routers placed at the edge each linking to 2 core routers closest to them. We use the graph-based "Disaster Management" category namespace from the Wikipedia database as our namespace [57]. Exploring 6 levels below that category, we obtain 489 categories and 732 relationships. If we seek to extract a set of hierarchies based on the approach in [16], we obtain 1,468 hierarchical names. We use the associated pages and files from the Wikipedia database (8,577 items total, 436 per category maximum, 17.49 on average per category) as the publications. We duplicated each content 60 times (514,620 publications) and ordered their publication randomly to load the network. While the namespace is static in our experiments, the publication workloads are dynamic and vary. Publications are generated using a Poisson distribution (to model human behaviors such as calling for, or offering to, help) with a *monotonically increasing arrival rate* over time (to model the increasing nature of such publications, as the disaster unfolds and more people become aware and get involved). We experiment with two example publication workloads: 1) moderate (average arrival rate varying from 1,500 pkt/s to 2,000 pkt/s) and 2) intense (arrival rate varying from 1,500 pkt/s to 3,500 pkt/s). We create 6 subscribers for each category (2,934 in total), distributed randomly on the 231 edge routers in the network. Eventually we generate 20,022,480 delivery events.

Experiments with moderate workload: We first consider the notification latency, to deliver a publication to all recipients. This reflects the impact of queuing in the network that arises from having to route through an RP, the selection of an appropriate number of RPs at the correct point in the network topology, adapting to the namespace and workload. We also look at the total network traffic to understand scalability.

We compare the performance of POISE with a number of alternatives. First, is the use of a strict hierarchical namespace (as in NDN/CNS). To be liberal to the hierarchical alternative, we avoid each subscriber having to subscribe to every name. Therefore, when there are multiple hierarchical names for a category, he subscribes to *any* one of the names. The publisher publishes to *all* the hierarchical names of the category. We also compare with having a single RP (no splitting), as well as a simple random splitting of the RP to one of the nodes in the network. The latter is used to demonstrate the need to use a near-optimal splitting of the RPs and load balancing in POISE.

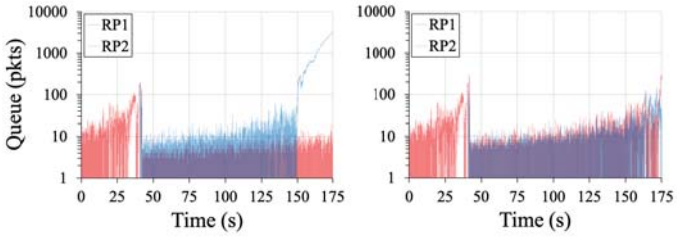
From the CDF of the notification latency in Fig. 9 (and the average reported in Table III), we see that due to the high workload on the RP caused by hierarchical names, the notification latency is excessive. Having only 1 RP (graph-1RP) as well as random splitting of RPs perform reasonably at lower loads (for rates $<$1700 pkt/s) and are even better than using hierarchical names at low loads. However, at higher workloads, random split and hierarchical names perform poorly compared to POISE as well as even having just one RP.

Fig. 8 shows the notification latency as the load gradually increases, for all the solutions. With a random split, Fig. 8b), the notification latency goes up very rapidly after the split, because the entire system is overloaded by packets sent back and forth between RPs. It is even worse than having a single RP, with no splitting (blue line in Fig. 8a). This shows the importance of a sensible partitioning algorithm. For the same workload, the latency of POISE (with METIS+Tabu, Fig. 8c) is dramatically better (by 2-orders of magnitude). As the load goes up, RP partitioning is triggered. For a short transient period, queuing causes a relatively small (compared to other alternatives) increase in latency. But congestion is immediately relieved by RP splitting and the latency drops back down. The maximum transient latency is 400 ms with POISE, compared to multiple seconds with other alternatives.

Next, we look at the total network traffic, summarized in Table III. Splitting the RP in POISE results in slightly higher traffic ($\sim$1%) due to the unicast between RPs, compared to having only a single RP. Yet by doing so, we avoid the significant latency impact of congestion. Random splitting of

TABLE IV: Comparison of METIS and POISE's partitioning.

Metric	METIS	POISE
Moderate workload		
Average latency (s)	0.018171	0.018147
Aggregated traffic (Gb)	491.242	492.392
Max Load - 2RP (#msgs)	1,321,220	1,230,533
Load imbalance - 2RP (#msgs)	360,693	633
Inter-RP messages - 2RP (#msgs)	136,571	314,139
Objective - 2RP	1,818,484	1,545,305
Intense workload		
First split time (s)	40.868	
Second split time (s)	150.388	174.252
Average latency in [0,170s] (s)	0.049686	0.020387



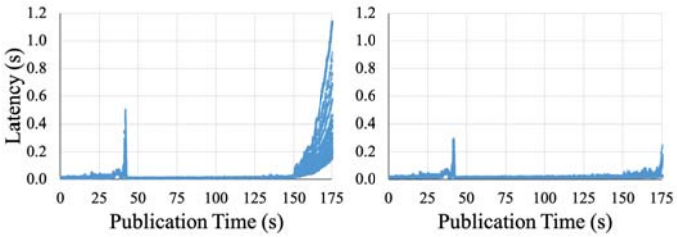
(a) METIS (b) METIS+Tabu (POISE)

Fig. 10: RP queue sizes for intense workload.

the RP performs much worse (*and* causing 27% more traffic compared to sensible splitting). In fact, if one were to just consider the relative increase in the amount of traffic because of RP splitting (compared to having just one RP and not having any RP splitting), then random splitting with 133.3 Gb of extra traffic results in 14.3 times more than POISE (9.3 Gb) in terms of extra traffic. Compared to the hierarchical solution, the graph-based solution of POISE reduces the amount of network traffic dramatically (by 75.9%) since we do not have to deal with the extra names and publications.

To dig a little deeper into the impact of the partitioning method used, we provide more detailed metrics in Table IV to compare the use of METIS and METIS+Tabu (POISE). For the moderate input workload, using METIS+Tabu leads to slightly (24μs) improved average notification latency (per delivery), while adding 0.002% total traffic. The reason for this better latency is better balance, and thus less queuing delay, even at the cost of slightly more traffic (just like a single RP having the least total traffic in Table III). The load metrics (in terms of # of messages) measure the RP load from the time of the split until the end of simulation (*i.e.* during the time the system has 2 RPs; labeled with '-2RP'). The table shows the values for the three terms in Eq.1. For METIS+Tabu, maximum RP load and load imbalance are significantly better, while for METIS, the # inter-RP messages is lower (leading to slightly less traffic). These combined, make METIS+Tabu's solution more balanced with a lower peak, as confirmed by the 'Objective' (sum of the above three terms), validating the effectiveness of our partitioning approach.

Experiments with intense workload: The benefit of METIS+Tabu over METIS is even more significant when we generate a higher intensity workload. The same publication trace (for ∼300s) was generated over a shorter duration (∼217s) by increasing the average inter-arrival rate. We also increase the RP splitting threshold. Table IV shows the time at which the first and the second RP splits occur; the first split is same for both (*i.e.*, 40.868s) while the second split occurs ∼24s later with METIS+Tabu compared to METIS (174s *vs.* 150s). The better balance with METIS+Tabu helps the single RP maintain the namespace for a longer time with lower dissemination latency; the same RP is used for 21% longer than the case of METIS. This is important, since the splitting procedure introduces protocol overhead (§V-B), with additional notification latency for a short period, as shown in Fig. 8c. Therefore, postponing splitting and reducing its frequency during the lifetime of the overall system is beneficial. However, if the split is postponed for too long, this latency would increase significantly, as seen in Table IV. For the period of [0,170s], the average notification latency of METIS is more than twice the latency of METIS+Tabu. Fig. 10 shows the instantaneous queue size of each RP in the two cases, for the period of [0,175s]. Most importantly, it shows the better balance between the two RPs in case of POISE (METIS+Tabu, Fig. 10b) than METIS (Fig. 10a). We also see that the size of the queue in RP2 for METIS goes up above 3,000 during that period, much larger than METIS+Tabu, which only goes up to 140 for the same time. As the figure shows, RP1's queue size is a little higher in POISE than METIS. However, the queue grows much more at RP2 in METIS than with POISE. This is the tradeoff that POISE makes, producing a better balance between the two RPs, thus helping prolong the need for splitting the RPs. The latency per publication for the intense workload is also shown in Fig. 11, indicating a much higher increase for METIS (Fig. 11a, seeing congestion after 150s) compared to METIS+Tabu (POISE, Fig. 11b, which stays low throughout, until 175s).



(a) METIS (b) METIS+Tabu (POISE)

Fig. 11: Notification latency for intense workload.

VII. Conclusion

We proposed POISE, an architecture for recipient-based pub/sub for disaster management, supporting free-form graph-based namespaces and automatic load splitting to eliminate traffic concentration based on a novel hybrid graph partitioning algorithm. Our simulation results show that POISE is efficient and scalable, compared to alternatives: its graph-based namespace outperforms the state-of-the-art hierarchical namespace of NDN [11]; its overall network architecture extends the recipient-based pub/sub framework of CNS [7]; and its partitioning outperforms the popular graph partitioner METIS [19].

VIII. Acknowledgements

This work was supported by the US Department of Commerce, National Institute of Standards and Technology (award 70NANB17H188) and US National Science Foundation grant CNS-1818971. We thank our shepherd, Andrei Gurtov, for his support and the reviewers for their valuable comments.

REFERENCES

[1] B. Winchel, "Las Vegas PD lauded for online response during mass shooting," Oct. 2017. [Online]. Available: https://www.prdaily.com/las-vegas-pd-lauded-for-online-response-during-mass-shooting/

[2] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps, "Content based routing with elvin4," in *in Proceedings of the Australian UNIX Users Group (AUUG2K)*, 2000.

[3] Y. Diao, S. Rizvi, and M. J. Franklin, "Towards an internet-scale xml dissemination service," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004.

[4] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni, "Tera: topic-based event routing for peer-to-peer architectures," in *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, 2007.

[5] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *International Conference on Broadband Communications, Networks and Systems*, 2010.

[6] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "Copss: An efficient content oriented publish/subscribe system," in *Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, 2011.

[7] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "Cns: content-oriented notification service for managing disasters," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016.

[8] A. Tagami, T. Yagyu, K. Sugiyama, M. Arumaithurai, K. Nakamura, T. Hasegawa, T. Asami, and K. Ramakrishnan, "Name-based push/pull message dissemination for disaster message board," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2016.

[9] M. Yuksel, K. K. Ramakrishnan, R. D. Doverspike, R. K. Sinha, G. Li, K. N. Oikonomou, and D. Wang, "Cross-layer failure restoration of IP multicast with applications to IPTV," *Computer Networks*, vol. 55, no. 9, pp. 2329–2351, 2011.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009.

[11] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J. Thornton, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[12] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "Mobilityfirst: a mobility-centric and trustworthy internet architecture," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, 2014.

[13] J. Seedorf, A. Tagami, M. Arumaithurai, Y. Koizumi, N. B. Melazzi, D. Kutscher, K. Sugiyama, T. Hasegawa, T. Asami, K. Ramakrishnan *et al.*, "The benefit of information centric networking for enabling communications in disaster scenarios," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015.

[14] I. Psaras, L. Saino, M. Arumaithurai, K. Ramakrishnan, and G. Pavlou, "Name-based replication priorities in disaster cases," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 434–439.

[15] A. Afanasyev, J. Shi *et al.*, "Nfd developer's guide," *Technical report, NDN-0021, NDN*, 2018.

[16] S. S. Adhatarao, J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "Comparison of naming schema in icn," in *Local and Metropolitan Area Networks (LANMAN), 2016 IEEE International Symposium on*, 2016.

[17] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "G-copss: A content centric communication infrastructure for gaming applications," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, 2012.

[18] A. Pinar and B. Hendrickson, "Partitioning for complex objectives," in *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, 2001.

[19] G. Karypis and V. Kumar, "MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0," http://www.cs.umn.edu/~metis, University of Minnesota, Minneapolis, MN, 2009.

[20] A. Zheng, A. Labrinidis, P. H. Pisciuneri, P. K. Chrysanthis, and P. Givi, "Paragon: Parallel architecture-aware graph partition refinement algorithm." in *EDBT*, 2016.

[21] X. Liu and A. A. Chien, "Traffic-based load balance for scalable network emulation," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. ACM, 2003.

[22] C. Curino, E. Jones, Y. Zhang, and S. Madden, "Schism: A workload-driven approach to database replication and partitioning," *Proc. VLDB Endow.*, vol. 3, no. 1-2, Sep. 2010.

[23] A. Bhatele, S. Fourestier, H. Menon, L. V. Kale, and F. Pellegrini, "Applying graph partitioning methods in measurement-based dynamic load balancing," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2011.

[24] Wikipedia, "Tabu search," https://en.wikipedia.org/wiki/Tabu_search, 2018.

[25] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "An information centric approach for communications in disaster situations," in *2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN)*. IEEE, 2014.

[26] M. Saunders, "Social media: California wildfires force thousands to evacuate," https://www.10news.com/news/social-media-california-wildfires-force-thousands-to-evacuate, Nov. 2018.

[27] M. Jahanian, Y. Xing, J. Chen, K. Ramakrishnan, H. Seferoglu, and M. Yuksel, "The evolving nature of disaster management in the internet and social media era," in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2018, pp. 79–84.

[28] H. M. Islam, D. Lagutin, A. Lukyanenko, A. Gurtov, and A. Ylä-Jääski, "Cidor: content distribution and retrieval in disaster networks for public protection," in *2017 IEEE 13th international conference on wireless and mobile computing, networking and communications (WiMob)*. IEEE, 2017, pp. 324–333.

[29] M. Jahanian, J. Chen, and K. Ramakrishnan, "Graph-based Namespaces and Load Sharing for Efficient Information Dissemination in Disasters," University of California, Riverside, Tech. Rep., 2019. [Online]. Available: http://www.cs.ucr.edu/~mjaha001/POISE-TR.pdf

[30] S. Mukherjee, F. Bronzino, S. Srinivasan, J. Chen, and D. Raychaudhuri, "Achieving scalable push multicast services using global name resolution," in *Global Communications Conference (GLOBECOM), 2016 IEEE*, 2016.

[31] J. Chen, M. Jahanian, and K. Ramakrishnan, "Black ice! using information centric networks for timely vehicular safety information dissemination," in *Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on*, 2017.

[32] "Wikipedia: Outline of knowledge," https://en.wikipedia.org/wiki/Portal:Contents/Outlines, 2019.

[33] R. Angles and C. Gutierrez, "Querying rdf data from a graph database perspective," in *European Semantic Web Conference*, 2005.

[34] R. Giugno and D. Shasha, "Graphgrep: A fast and universal method for querying graphs," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, 2002.

[35] S. Flesca and S. Greco, "Querying graph databases," in *International Conference on Extending Database Technology*, 2000.

[36] L. Cardelli, P. Gardner, and G. Ghelli, "A spatial logic for querying graphs," in *International Colloquium on Automata, Languages, and Programming*, 2002.

[37] "Kubernetes," https://kubernetes.io/, 2019.

[38] D. Di Sarli and F. Geraci, "Gfs: A graph-based file system enhanced with semantic features," in *Proceedings of the 2017 International Conference on Information System and Data Mining*, 2017.

[39] A. E. Feldmann and L. Foschini, "Balanced partitions of trees and applications," *Algorithmica*, vol. 71, no. 2, 2015.

[40] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, 1998.

[41] ——, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, 1998.

[42] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[43] J. Nishimura and J. Ugander, "Restreaming graph partitioning: simple versatile algorithms for advanced balancing," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.

[44] E. Rolland, H. Pirkul, and F. Glover, "Tabu search for graph partitioning," *Annals of Operations Research*, vol. 63, no. 2, 1996.

[45] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning," *Operations research*, vol. 37, no. 6, 1989.
[46] B. Uçar and C. Aykanat, "Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies," *SIAM Journal on Scientific Computing*, vol. 25, no. 6, 2004.
[47] I. Moulitsas and G. Karypis, "Partitioning algorithms for simultaneously balancing iterative and direct methods," Minnesota Univ Minneapolis Dept of Computer Science, Tech. Rep., 2004.
[48] R. H. Bisseling and W. Meesen, "Communication balancing in parallel sparse matrix-vector multiplication," *Electronic Transactions on Numerical Analysis*, vol. 21, 2005.
[49] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, August 2006.
[50] M. J. Donahoo and E. W. Zegura, "Core migration for dynamic multicast routing," Georgia Institute of Technology, Tech. Rep., 1995.
[51] Ying-Dar Lin, Nai-Bin Hsu, and Chen-Ju Pan, "Extension of rp relocation to pim-sm multicast routing," in *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 1, 2001, pp. 234–238 vol.1.
[52] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang, "Ndns: A dns-like name service for ndn," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, 2017.
[53] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM*, 2004.
[54] "Poise simulator," 2019. [Online]. Available: https://github.com/SAIDProtocol/NetworkSimulator
[55] GDdata, "Graph Drawing," http://www.graphdrawing.org/data.html, 2018.
[56] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.
[57] Wikipedia, "Category:Disaster management," https://en.wikipedia.org/wiki/Category:Disaster_management, 2018.