CoNICE: Consensus in Intermittently-Connected Environments by Exploiting Naming With Application to Emergency Response

Mohammad Jahanian and K. K. Ramakrishnan, *Fellow, IEEE, ACM*

***Abstract*— In many scenarios, information must be disseminated over intermittently-connected environments when the network infrastructure becomes unavailable, *e.g.*, during disasters where first responders need to send updates about critical tasks. If such updates pertain to a shared data set, dissemination consistency is important. This can be achieved through causal ordering and consensus. Popular consensus algorithms, *e.g.*, Paxos, are most suited for connected environments. While some work has been done on designing consensus algorithms for intermittently-connected environments, such as the One-Third Rule (OTR) algorithm, there is still need to improve their efficiency and timely completion. We propose CoNICE, a framework to ensure consistent dissemination of updates among users in intermittently-connected, infrastructure-less environments. It achieves efficiency by exploiting hierarchical namespaces for faster convergence, and lower communication overhead. CoNICE provides three levels of consistency to users, namely replication, causality and agreement. It uses epidemic propagation to provide adequate replication ratios, and optimizes and extends Vector Clocks to provide causality. To ensure agreement, CoNICE extends OTR to also support long-term network fragmentation and decision invalidation scenarios; we define local and global consensus pertaining to within and across fragments respectively. We integrate CoNICE's consistency preservation with a naming schema that follows a topic hierarchy-based dissemination framework, to improve functionality and performance. Using the Heard-Of model formalism, we prove CoNICE's consensus to be correct. Our technique extends previously established proof methods for consensus in asynchronous environments. Performing city-scale simulation, we demonstrate CoNICE's scalability in achieving consistency in convergence time, utilization of network resources, and reduced energy consumption.**

***Index Terms*— Consensus, Delay-Tolerant Networks (DTN), disaster management, information-centric networks.**

I. Introduction

AS THE world becomes more and more dependent on network connectivity, it is also important to be resilient to situations when connectivity is intermittent. A pertinent example especially in the recent years, which we consider in this paper as a use case, is emergency response, where the networking infrastructure, *e.g.*, cellular access, becomes damaged and is thus unavailable [1]. The design and use of protocols for information dissemination that tolerate intermittent connectivity [2] become important. To address such scenarios, Opportunistic Networks tolerate a disconnected topology graph with mobile nodes [3]. They leverage Device-to-Device (D2D) [4] message exchanges in mobile encounters between nodes, as in Delay-Tolerant Networks (DTN) [2], without relying on network infrastructure or the availability of an end-to-end path.

Ensuring the consistency of updates that are disseminated among participants and the 'rest of the world' is important. It is challenging to support distributed applications, such as the ones where multiple users are applying changes to a shared common database, in intermittently-connected environments. Continuing with the emergency response scenario, an example of such distributed application, one which has gained a lot of attention recently, is geo-tagging of key locations such as disaster-impacted sites. First responders involved in search and rescue missions may mark on a map on their smart phones of such locations and have to be updated across all the devices of group members as well as deliver a reliable, consistent view to incident commanders and others that require situational awareness. Many algorithms and techniques to ensure consistency have been proposed [5]. Causal consistency ensures updates get processed at users in accordance with their causal relations [6]. Causal ordering provides a 'moderate' degree of consistency [7] , which is stronger than best-effort out-of-order delivery, as it orders "orderable" updates. It is weaker than agreement-based total order delivery, as it is ambiguous when it comes to ordering "un-orderable" updates.

Consensus methods, on the other hand, ensure agreement and strong consistency. There are many proposals, most notably Paxos [8]. Consensus is an important distributed algorithm with many applications such as in datacenters [9], banking systems [5], and Blockchains [10]. An issue with the applicability of most of these consensus solutions is that they are suited for connected and (partially) synchronous environments. This has led to the design of consensus algorithms and protocols for disconnected and asynchronous environments, such as Paxos/LastVoting [11] and One-Third Rule (OTR) [12] algorithms. However, these solutions assume that the majority of users have "good periods" [13] throughout

Manuscript received December 24, 2020; revised June 24, 2021, September 26, 2021, and January 1, 2022; accepted February 6, 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor H. Shen. This work was supported in part by the U.S. Department of Commerce, National Institute of Standards and Technology (NIST), under Award 70NANB17H188; and in part by the U.S. NSF under Grant CNS-1818971. *(Corresponding author: Mohammad Jahanian.)*

Mohammad Jahanian was with the Department of Computer Science and Engineering, University of California, Riverside, CA 92521 USA. He is now with Aruba, a Hewlett Packard Enterprise Company (e-mail: mjaha001@ucr.edu).

K. K. Ramakrishnan is with the Department of Computer Science and Engineering, University of California, Riverside, CA 92521 USA (e-mail: kk@cs.ucr.edu).

Digital Object Identifier 10.1109/TNET.2022.3156101

the whole network, not supporting scenarios with long-term fragmentation in which two isolated groups of users conduct independent consensus sessions for the same "ballot" and decide differently. Also, they are often too slow to converge, as they need to involve the whole network due to lack of systematic topic-based clustering of users.

The advance of information-centric paradigms [14], inspired by the content-oriented network usage of today, has led to the proposal and design of widespread in-network naming frameworks for information dissemination [15]–[17], that is better organized, showing that with a proper naming schema [18], we can achieve better accuracy and more scalable dissemination in terms of reducing end user and network load, compared to the current address-oriented networking paradigms. Furthermore, the notion of naming provides a location-independent forwarding capability with mobility [19], especially in highly dynamic environments where users' point of attachments keeps changing very frequently.

In this paper, we propose CoNICE (**Co**nsensus in **N**ame-based **I**ntermittently-**C**onnected **E**nvironments), a framework for consistent dissemination of updates of a shared database among mobile users, in an intermittently-connected environment. We assume no networking infrastructure, no geographical routing or synchronized physical clocks. CoNICE uses graph-based naming [20] to systematically divide the physical space (through region-ing) and the consensus space (through user subscriptions) into hierarchically structured subsets, optimizing the consensus participation to get higher completion rate and faster completion times. CoNICE extends existing name-based information dissemination schemes (such as [15], [20]) by ensuring consistency, and resiliency in infrastructure-less environments. CoNICE is inherently failure-resilient, where disconnection is not just a corner case scenario, but is rather a common case. Inspired by the multi-level consistency requirements provided by cloud and database systems [21], [22], CoNICE provides the coexistence & flexibility of the following three incremental consistency levels for the network: replication (weakest consistency, lowest complexity), causality, and agreement (strongest consistency, highest complexity). All these consistency levels are integrated with a topic-based hierarchical naming schema, through Name-based Interest Profiles (NBIP). The consensus protocol of CoNICE (running on top of D2D propagation protocols), provides users with a strongly-consistent view that respects both agreement and causality. CoNICE extends OTR with naming and decision invalidation handling procedures, for a total and causal ordering of updates. Its decision invalidation helps with overcoming the consensus property violations in case of long-term physical fragmentation in the network, mainly since we define the complementary notions of local and global consensus sessions, pertaining to being within and across fragments, respectively. This invalidation is an important novelty of CoNICE, as it allows the consistency to be achieved even when many fragmented users connect after a long time (such as users from disjoint and remote shelters, in the aftermath of a natural disaster). Existing proof methods for asynchronous consensus algorithms assume "good periods" throughout the whole network (such as in [13]). We take a step further by assuming that these good periods are within network fragments which are disconnected for long periods, thus expanding the scope of the asynchronous consensus problem, which is also suitable for our application scenario.

A key novelty of CoNICE is its integration of consistency and dissemination through naming. In other words, the graph-based namespace works as a common interface across the modules that take care of multiple levels of consistency, as well as the protocols for content dissemination throughout the users in the network. The benefit of this use of naming is twofold: 1) it enhances the relevance of information dissemination (*i.e.*, recipients can identify relevant content with respect to their interests) in a decoupled pub/sub manner (*i.e.*, publishers and subscribers do not need to keep track of each other); 2) it enhances the degree of information consistency among relevant users (*i.e.*, optimized by related name-based groups, consensus can be reached faster, and to a higher degree). We will demonstrate these benefits through our results.

The major contributions of the paper are: 1) A framework for consistent information dissemination in intermittently-connected environments, considering the important case of emergency response (our source code and data are available [23]); 2) Enabling different incremental consistency levels (replication, causality, and agreement) for information updates in intermittently-connected networks; 3) A systematic coupling of information flow organization with various consistency preservation procedures, using naming graphs; 4) Extending the OTR consensus with a protocol that leverages naming and supports recovery from invalidated decisions; 5) A rigorous proof of CoNICE's consensus protocol using the Heard-Of model formalism, extending the classic OTR proof to cases supporting long-term fragmentation and decision invalidations. 6) Simulation results that show our enhancement leads to a higher degree of agreement among users, with lower overhead; 7) Simulation experiments that demonstrate how CoNICE is resilient to long-term fragmentation scenarios through an effective procedure for decision invalidation, extending existing work on asynchronous consensus solutions.

II. Background and Related Work

Propagation in Intermittently-Connected Environments. There have been a number of works on information propagation in intermittently-connected networks [24]. Generally, these solutions rely on nodes to store, carry, and forward messages [2]. Most solutions rely on nodes taking advantage of opportunistic encounters to exchange messages (*i.e.*, gossiping), typically with high message delivery latency due to disconnections [25]–[27]. Methods such as Bubble Rap [28], dLife [29], SCORP [30], and EpSoc [31] use social data regarding human interactions as the basis of such routing predictions. We use Epidemic Routing [25] in this paper because of its simplicity for DTNs and the fact that it requires minimum assumptions about network (no path/geography/social-connection based decisions) which suits our scenarios, has a high delivery ratio, achieves lower delays (relatively), and is especially suitable for broadcast-oriented messaging [24] (although we can replace it with the some of the other methods

mentioned if additional assumptions are reasonable and can be accommodated). Epidemic routing uses message buffers and performs store-and-forward [25]. Apart from its benefits, it is observed that epidemic routing has high overhead [24]. We enhance it with the use of naming, to reduce load.

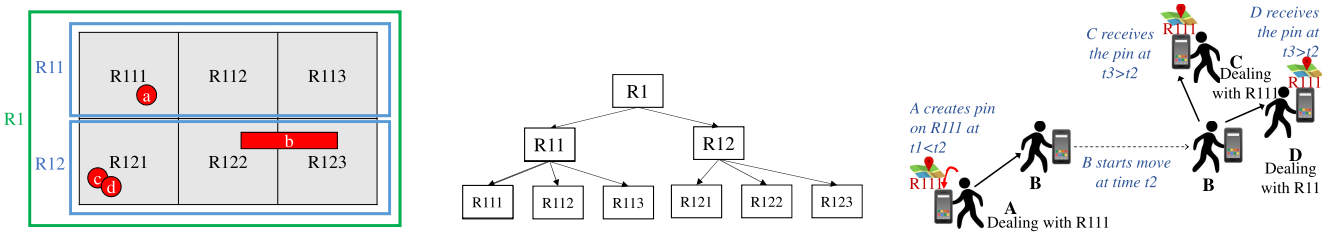
Causal Consistency. Causal consistency is a popular consistency model which ensures ordering of events (*e.g.*, network messages) based on their causal relationship. Works such as [32] propose the use of physical clocks for ordering. However, physical clocks may have skews. The protocol for clock synchronization may involve significant overhead, especially in a disconnected environment. Scalar logical clock [6] defines the "happened before" relation. A message is said to be *causally delivered* at a recipient user, if all the causal prerequisites of that message have been delivered at the user too [5]. The Vector clock method [33], [34] ensures causal ordering using vectors carried as history in each message, that represent the sender's current state relative to every other users' progress. Work in [35] proposes differential clocks as an optimization to vector clocks, only sending vector differences. [36] proposes that explicitly specifying causality, by sender, helps with scalability. Work in [37] proposes a method for group causal ordering, and enabling causal delivery to multiple groups of interested users. We use the notion of vector clock but extend it to enable selectiveness through hierarchically-structured naming and a reactive mode for faster causal delivery, and capture both implicit and explicit causality.

Consensus and Strong Consistency. There has been a great deal of work on consensus, the most prominent of which is Paxos [8]. Paxos achieves agreement among a number of networked nodes, by election of a leader, majority voting, and deciding on a value, through a number of rounds. Raft [9] implements Paxos, designed for strong consistency in log replication among servers in a cluster. It has been observed that such solutions work well in connected environments. The reason for that is they assume synchronous or partially synchronous systems [13]; that it has an upper bound on latency of message delivery between any two nodes. That assumption is challenged in asynchronous systems, where there may be many failures, leading to non-terminating consensus procedures. The popular method to address this challenge is "augmenting" asynchronous systems with external devices, most notably Failure Detectors using oracle nodes [38]. While failure detectors make consensus more failure resilient, they are limited to node failures only, rather than link failures. Therefore, they are not particularly efficient in a highly-transient DTN environment, such as ours, as they lead to too many timeouts when a mobile node goes away for some time even if the node has not crashed. To address this, the Heard-Of model [13] proposes a *benign fault* model that tolerates link failure, and proves that the consensus algorithms Paxos/LastVoting (P/LV) [11] and One-Third Rule (OTR) [12] can tolerate loss and be suitable for intermittently-connected and mobile environments. The model demonstrates that rather than assuming eventual synchrony, it is more realistic to assume "good periods" in asynchronous systems, *i.e.*, an epoch in which nodes can hear of each other (receive their messages). Work in [39] proves the one-third rule to reach correct consensus and possibly finish in one round, as long as no more than one third of the consensus participants crash. Another benefit of OTR over P/LV is that it is coordinator-less, and thus does not need to have the overhead and complexity of leader election. We build on OTR, enhancing it with an integration of naming, to optimize storage and energy consumption as well as faster consensus completion time. We add support for cases where decisions need to be invalidated *e.g.*, due to long-term network fragmentation. To that end, we enhance OTR by relaxing its assumption of good periods across the whole network. Instead, we consider these periods are only within temporarily disconnected network fragments. Thus, we tolerate long-term fragmentation in the network.

Name-based Information Dissemination. The use of network naming for systematic organization of information for better dissemination efficiency has been introduced as the integral part [40] of the Information-Centric paradigms, such as in Named Data Networks (NDN) [15]. Work in [41] provides name-based DTN-like dissemination frameworks. However, extra steps are needed for ensuring consistent dissemination. Some works [12] propose the use of interest profiling for selective gossiping. We extend their ideas to implement the content-oriented graph-based naming for profiling as well as proposing a flexible multi-level profiling for various consistency levels. Methods such as NDN Sync [42] and Secure Scuttlebutt [43] propose log replication consistency in name-based intermittently-connected environments. However, they only guarantee causal consistency, but do not provide strong consistency or total ordering, which are important when dealing with multi-user updates on a single, shared database. Naxos [44] proposes a name-based version of Paxos for NDN. However, Naxos only supports request/response pull-based communication pattern, and assumes connected environments with centralized orchestration. We integrate name-based publish/subscribe push-based dissemination patterns, and aim at ensuring strong total order consistency by supporting consensus in dynamic intermittently-connected environments.

III. Overview of CoNICE

Emergency Response Scenario. During, and/or immediately after a disaster (*e.g.*, a natural disaster such as a hurricane, earthquake, or wildfire), many teams of first responders are mobilized, especially for search and rescue. Such teams move around different neighborhoods and mark locations with various codes (based on marking standards of the Federal Emergency Management Agency [45]). This was used in several disasters, such as Hurricane Katrina [46]. We extend the manual marking system with a map-based application for more effective coordination and information dissemination among first responders. We outline an example use case for emergency response where first responders seek to individually update map tags on their devices and then need to arrive at a consistent, coherent view across users as they opportunistically connect with each other. The map in CoNICE is made up of a base layer and data layer, as shown in the example in Fig. 1(a). The base layer is the map background, available offline to each user. Informed by the geography pertaining



(a) Region-ed map: base layer and data layer (b) Namespace pertaining to the map in Fig. 1(a) (c) A scenario overview

Fig. 1. Example namespace and scenario.

to the map, it is divided into hierarchically-structured regions (*e.g.*, county, city, *etc.*). For example, region $R11$ is part of $R1$, and is made up of $R111$, $R112$, and $R113$. This hierarchical structure is captured in a namespace, as shown in Fig. 1(b). Users dynamically create updates on the map (*i.e.*, pins or other shapes with data on them), which updates the map data layer; *e.g.*, update 'a' as a point in $R111$, or 'b' as a shape spanning regions $R122$ and $R123$ in Fig. 1(a). Users create and are interested in receiving updates related to the regions they are dealing with (or to a part of the region they are interested in). As shown in Fig. 1(c), the environment we consider is one without infrastructure-based communication and users rely on D2D [4] communications, with frequent disconnections. Users are equipped with mobile devices capable of D2D wireless communication (*e.g.*, Bluetooth or WiFi-Direct), and have the CoNICE application on their device. In the example scenario (Fig. 1(c)), user A (a first responder) creates a pin on region $R111$ and propagates it at time $t1$. Users C and D, who are both interested in $R111$ (through subscribed interest in regions $R11$ and $R111$ respectively), have no path to A at $t1$. However, thanks to user B moving between the two fragments and acting as a mule doing store-carry-and-forward [2], the update gets propagated to C and D, and they can add it to their view of the map. Different users can create updates (on top of their accumulated 'view' of the data layer) and disseminate them at any time, without any coordination. Considering the example in Fig. 1(a), let us assume update 'a' has been (at least partially) propagated among interested recipients of region $R111$. If now two different users simultaneously create new updates that modify 'a', they would both consider that as "the next update in $R111$". This may cause discrepancy in the order in which different users apply the updates. This is an important issue that we need to address for the effectiveness and correct functionality of CoNICE in critical scenarios, such as disaster management. Thus, our primary goal is to make sure all the users converge to a consistent view of the updates on map data layer as much as possible.

CoNICE Overview. An overview of the architecture of CoNICE is depicted in Fig. 2. It consists of the integration of *multi-level consistency* and *multi-level naming*. There are three incremental levels of consistency. Consistency level 0, namely *Replication*, suggests how much of the generated updates have been delivered to individual users. The *Gossiping* component in each user's device is responsible for this function, using Epidemic Routing [25]. Consistency level 1, namely *Causality*, ensures that orderable updates are applied according to their causal relationships and precedence. This

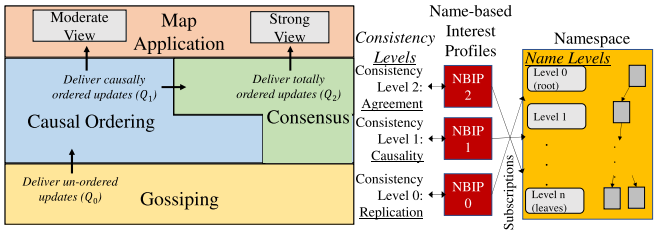


Fig. 2. Architecture overview of CoNICE.

is provided by the *Causal Ordering* component, which provides a moderately-consistent view (*Moderate View*) of the map to the user in the application. CoNICE uses a Vector Clock-based approach [33], [34], extended by a selective and reactive repair mode for causal ordering. Consistency level 2, namely *Agreement*, deals with achieving agreement between different users' views, even for un-orderable updates. The *Consensus* component enables this, and provides the user with a strongly-consistent view (*Strong View*). For this component, CoNICE implements a solution based on the One-Third Rule (OTR) consensus algorithm [12], extending it by supporting selective participation and decision invalidations for highly fragmented and intermittently connected scenarios. Every user is equipped with a single, unified namespace; a hierarchically structured graph pertaining to the regions in the map. This namespace drives the various consistency level components, achieved by *Name-Based Interest Profiles* (NBIP) in CoNICE. There is a NBIP for every consistency level, each pointing (as a subscription) to a particular subset of the namespace. The use of NBIPs allows the various components to achieve better efficiency and accuracy in dissemination. More foundational details on naming and consistency levels are in [47].

While we recognize security is important to make CoNICE's design robust and usable, we have to address it in detail in a separate work complementary to this paper. Here we address the basic protocol of CoNICE and its properties. To ensure authentication and integrity, we can use hash chains, similar to [43]; it complements CoNICE's design, since it is based on sequential updates, each update depending on (and cryptographically linked to) the previous one. Further, to ensure fine-grained access control, attribute-based encryption can be leveraged, similar to [48]; it fits well with CoNICE, since it incorporates a namespace, and each user's access privileges corresponding to their role-based subscription. Thus, CoNICE can prevent malicious attacks such as impersonation and forgery, via information-centric security [49], which secures

content itself, rather than the delivery channel. Additionally, Machine Learning-based solutions, such as those proposed in [50] can be used for malware detection and privacy protection.

IV. Protocols for Consistent Dissemination

We describe the protocols for different consistency levels.

A. Gossiping Protocol

We use epidemic propagation [25] for gossiping; note that it can be replaced with other information propagation methods given particular system assumptions, as long as they allow anycast propagation. Our assumption in CoNICE is that each user has a unique user ID (*UID*), which can be the mobile device's IMEI or a number provided by the CoNICE application at the time of installation. In CoNICE, each message has a *tag*, specifying its type. Each message has a message ID (*MID*) which is used to uniquely identify it in the network. Users buffer messages for epidemic propagation indexed by their MIDs. Users can create, relay (*i.e.*, store, carry, and forward), and receive messages. CoNICE makes this propagation selective via interest profiling, namely NBIP0 for gossiping. Typically, benevolent data mules help with relaying any message, regardless of what they are about, while other users (*e.g.*, first responders) can have a more fine-grained NBIP0 and only receive and relay messages matching their interest, while discarding others. Updates contain the ID of the region they belong to (R_B) and the (set of) regions they cover (R_C). R_B is integral across all levels, while R_C is only used at level 0 (*i.e.*, can be compared against NBIP0), and its purpose is to increase coverage. A user receiving a message based on its R_C, the R_B of which he is not subscribed to, can be considered to have subscribed to the R_B in the namespace hierarchy, to be able to also participate in its level 1 & 2 procedures. This epidemic propagation can potentially cause excessive overheads, causing two challenges: 1) a message may keep travelling too many hops, causing unnecessary network & queuing overhead (*e.g.*, a user may receive the same content many times from multiple paths), and 2) too many messages will stay in user device buffers for excessive periods, causing unnecessary storage and processing overhead. To remedy, CoNICE uses hop count limits and cleaning buffers of obsolete messages (similar to [12]).

B. Causal Ordering Protocol

Different updates that belong to the same region can potentially depend on each other. As an example, in Fig. 1(a), update 'd' may depend on 'c', as they both belong to $R121$, and the creator of 'd' has seen 'c' (may be the same creator) ; *e.g.*, 'd' may remove some data that 'c' has added, or modify the information provided by 'c' about a particular disaster site. These dependencies need to be specified and considered both when it comes to creating and publishing updates, and receiving and processing them. The causal ordering component in CoNICE takes care of this, which helps with consistency level 1 (causality). We restrict Lamport's "happened before" relation [6] to only messages that belong to the same region, calling it "happened before in the same region", to capture causality. This is possible since the region ID is already carried in updates in CoNICE. Formally, we define it as the following:

Definition 1: For updates u_1 and u_2, belonging to the same region, u_1 causally precedes u_2 ($u_1 \rightarrow u_2$) if and only if one of the following three conditions are true: 1) Some user U creates and publishes u_1, and then creates and publishes u_2 (*FIFO order*), 2) Some user U receives and delivers u_1, and then creates and publishes u_2 of the same region (*local order* or *network order*). 3) There is an update u_3 such that $u_1 \rightarrow u_3$ and $u_3 \rightarrow u_2$ (*transitivity*).

The procedure for update creation is shown in Alg. 1. An update message in CoNICE is of the form shown in line 13. The update ID ($UpID$) consists of UID_A (user ID of update creator A), R_B (the region the update 'belongs to'), $seqNum$ (sequence number), R_C (set of leaf-node regions 'covered' by the update), and $UpID_R$ (set of references for this update, which we explain later). The $data$ element contains the map-related instruction for the update (*e.g.*, "mark house #1 as searched" or "need teams at building #2"). It is important to also include dependency information in the update, on top of basic gossiping, so that recipient will be able to order the received updates (which can be potentially delivered out-of-order), and achieve a consistent view across different mobile users. CoNICE updates contain dependencies in two ways: *implicit dependency* and *explicit dependency*. Implicit dependency pertains to the dependency of the update on its creator's previous updates on the same region (thus ensuring the FIFO ordering [5]). For a new update in region R_B, the creating user looks for the highest $seqNum$ it has used for R_B so far, and assigns the next number to the update (line 8). Explicit dependency pertains to the dependency of the update on other users' previous updates on the same region (lines 9–12), that creator A has already causally delivered to higher layers The helper function $creators()$ (line 9) traverses through A's level 1 queue and returns all the users who have contributed to it. (thus ensuring the local ordering [5]). For each $<user,R_B>$ pair, only the update ID with the highest sequence number is picked (line 10). To further reduce the message overhead, all those updates u in $UpID_R$ that precede an update u' existing in $UpID_R$, are removed from the references list (line 12). As a result, the reference list in CoNICE updates will be more compact than the full vector of VC, and also relieves users from having to maintain a global vector of every user in the network. Finally, the created update will be published by sending it to the gossiping module, and will be added to the creating user's level 1 queue (lines 13–16).

The procedure for handling the receipt of updates and causally delivering them is described in Alg. 2. The processing of the incoming update u only proceeds at user A if the R_B 'belongs' to its NBIP1 (line 13). Pending updates that get satisfied, will be added to Q_1 (lines 14–15), and all (implicit and explicit) missing prerequisites of u will be collected in $missing$ (lines 16–17). If there are no missing prerequisites, u will be causally delivered and applied to its 'Moderate View' (line 18). In case of outstanding missing prerequisites, the VC algorithm typically waits till they are received. In a

Algorithm 1 Update Creation With Causality

```
1  input:
2      R_B: belongs-to region; R_C: set of regions covered; data: update data
3  initialization:
4      UID_A ← id of this user A
5      M_A ← set of updates created by A
6      M_Q1 ← set of all updates at level 1 queues at A
7      UpID_R ← {} /* set of reference updates to be
           included                                          */
8  seqNum ← nextSeqNum(R_B, M_A)
9  foreach user B in creators(M_Q1) do /* identify references */
10     UpID_R ← UpID_R ∪ latestUpdate(B)
11 foreach u ∈ UpID_R do /* make reference list more compact
   */
12     if ∃u' ∈ UpID_R : u → u' then UpID_R ← UpID_R − {u}
13 msg ← ⟨UPDATE, UID_A, R_B, seqNum, R_C, UpID_R, data⟩
14 publish msg /* send to gossiping module                  */
15 M_A ← M_A ∪ msg
16 M_Q1 ← M_Q1 ∪ msg
```

Algorithm 2 Update Receiving and Causal Ordering

```
1  input:
2      R_B, R_C, data: as in Alg. 1; UID_C: user id of the update creator;
3      seqNum: update sequence number; UpID_R: set of update references;
4      UID_R: user id of requestor in the response msg
5  initialization:
6      UID_A ← id of this user A
7      M_Q0 ← set of all updates at level 0 queue at A
8      M_Q1 ← set of all updates at level 1 queues at A
9      missing ← {} /* update IDs of missing prerequisites
           */
10 Upon receive
   (msg=⟨UPDATE, UID_C, R_B, seqNum, R_C, UpID_R, data⟩) do
11     procUpdate(UID_C, R_B, seqNum, R_C, UpID_R, data)
12 Procedure procUpdate(UID_C, R_B, seqNum, R_C, UpID_R, data)
13     if R_B ∈ NBIP1_A then
14         foreach u' ∈ M_Q0 ∧ u' ∉ M_Q1 do
15             if dependencies(u') satisfied then M_Q1 ← M_Q1 ∪ {u'}
16         missing ← missing ∪ missingImplicit(u, M_Q1)
17         missing ← missing ∪ missingExplicit(u, M_Q1)
18         if missing = {} then M_Q1 ← M_Q1 ∪ {u}
19         foreach UpID_i ∈ missing do publish⟨REQUEST, UpID_i⟩
20 Upon receive (msg=⟨RESPONSE, UID_R, UID_C, R_B, seqNum,
   R_C, UpID_R, data⟩) do
21     procUpdate(UID_C, R_B, seqNum, R_C, UpID_R, data)
22     if UID_R = UID_A then cancel msg
```

Algorithm 3 Consensus: Contributions

```
1  input:
2      R_S: region for this session S; s_S: slot number to be decided for S;
3      n_S: user A's estimation of population for S
4  initialization:
5      Q_1^A ← user A's current level 1 queue
6      Q_2^A ← user A's current level 2 queue
7      UID_A ← id of this user A
8      contribs_s ← {}      /* contributions multiset at A */
9      dec_S = ⟨DECISION, UID_D, R_S, s_S, a_D, n_D, v_D⟩ ← {}
10     solved_S ← false /* as dec_S is empty initially      */
11     v_I ← Q_1^A(R_S, s_S) /* Noop if null                */
12     if v_I ≠ Noop then startAttempt(1, 1, v_I)
13 Procedure startAttempt(a, r, v)
14     v_S ← v
15     a_S ← a
16     startRound(a_S, r)
17 Procedure startRound(r)
18     r_S ← r
19     publish msg=⟨CONTRIBUTION, UID_A, R_S, s_S, a_S, r_S, n_s, v_s⟩
20     contribs_S ← contribs_S ∪ msg
21 Upon receive (msg=⟨CONTRIBUTION, UID, R_S, s_S, a, r, n, v⟩) do
22     if R_S ∉ NBIP2_A then cancel msg
23     switch a do
24         case a > a_S
25             foreach m ∈ contribs_S do cancel and delete m
26             n_S ← max(n_S, n)
27             contribs_S ← {msg}
28             startAttempt(a, r)
29         case a < a_S  publish D_S
30     if solved_S then
31         publish D_S
32         cancel and delete msg
33     switch r do
34         case r > r_S
35             foreach m ∈ contribs_S do cancel and delete m
36             n_s ← max(n_S, n)
37             contribs_S ← {msg}
38             startRound(r)
39         case r < r_S  cancel and delete msg
40         case r = r_S
41             contribs_S ← contribs_S ∪ msg
42             n_s ← max(n_S, n)
43             if |contribs_S| > (2/3) × n_S then
44                 v_S ← smallest most frequent non-Noop in contribs_S
45                 if all equal to V in contribs_S excluding Noop then
                       decide(R_S, s_S, a_S, v_S)
```

disconnected environment with gossiping, this may lead to starvation and indefinite waiting, since "gossips may die out" [26]. To remedy this, CoNICE adds a reactive recipient-driven procedure of requesting for those missing updates (line 19). The $REQUEST$ message identifies the update ID requested for, and the requester's ID (UID_R). Any user, not necessarily the creator of the update, who has that update buffered, can respond with a $RESPONSE$ message, sent for the requester. When receiving a response, user A processes it in a similar manner to a normal $UPDATE$ message, with one difference that if the response was meant for A, A will cancel the update and not propagate it in the network further (lines 20–22). CoNICE ensures the following key property (proof in §V):

Property 1: Causal Order of Moderate View. If user A applies (and delivers) update u to its moderate view, then A must apply every update causally preceding u before u.

Proof of Property 1: We can prove this property using induction. *Basis:* If A applies (to its moderate view) no updates, the property holds. If it applies only one update u_1 belonging to R_B, per Alg. 1 and 2, u_1 had no implicit references (*i.e.*, first update created by its creator B for R_B) and no explicit references (*i.e.*, no other user C has created an update for R_B that B had applied before creating u_1). *Inductive step:* Let us assume A has applied n updates $u_1, \ldots, u_n$, preserving the causal ordering property. An additional update u_{n+1} will only be applied at A, if and only if all causal prerequisites of u_{n+1} are already in $u_1, \ldots, u_n$ and there are no missing updates (per lines 14–19 in Alg. 2), thus ensuring Property 1 holds.

C. Consensus Protocol

CoNICE provides a consensus procedure with the goal of achieving agreement, so that users (*e.g.*, first responders) have the same consistent 'Strong View' of the situation (*e.g.*, map). The consensus solution in CoNICE builds on the One-Third Rule (OTR) algorithm [12]. We extend OTR in several ways, mainly with regards to naming and decision invalidations. The naming integration in CoNICE, makes sure all the interested users (even with overlapping interests) are involved in every consensus session relevant to them, which also systematically reduces the consensus participants to the interested ones, helping with faster reaching of decisions. CoNICE's decision invalidation procedures make sure to repair decisions if long-term fragmentation cases happen in the network, and also if the total and causal order of the final strong view are violated even after the OTR-based agreement is reached.

The initialization and contribution procedures of CoNICE's consensus are described in Alg. 3. Each consensus session is associated with a region-slot pair ($<R_S, s_S>$), deciding the value v (*i.e.*, the update to be placed at the slot) to be inserted to $Q_2(R_S, s_S)$. To avoid scheduling complexities and

Algorithm 4 Consensus: Decisions

```
1  Procedure decide(UID, R_S, s_S, a, n, v)
2      if ¬solved_S then
3          if ∃s' ≠ s_S ∧ Q_2^A(R_S, s') = v then
               /* conflict with earlier existing decision */
4              if v_I = v then startAttempt(a_S + 1, 1, Noop)
5              else startAttempt(a_S + 1, 1, v_I)
6          solved_S ← true
7          v_S ← v
8          foreach m ∈ contribs_S do cancel and delete m
9          contribs_S ← {}
10         publish msg = ⟨DECISION, UID_A, R_S, s_S, a, n, v_S⟩
11         D_S ← msg
12     else                                   /* need to invalidate */
13         if msg ≠ D_S then
14             if a = a_D then
15                 if n > n_D then v'_D ← v
16                 else if n < n_D then v'_D ← v_D
17                 else if n = n_D then
18                     if UID ≥ UID_A then v'_D ← v
19                     else if UID < UID_D then v'_D ← v_D
20                 decide(max(UID_D, UID), R_S, s_S, a, n, v'_D)
21             if a > a_D then
22                 startAttempt(a, 1, v_I)
23                 decide(UID, R_S, s_S, a, n, v)
24         Q_2^A(R_S, s_S) ← v
25         foreach v' ∈ Q_2^A(R_S) that violates causality with v do
26             reorder locally through deterministic sort
27 Upon receive (msg = ⟨DECISION, UID, R_S, s_S, a, n, v⟩)do
28     if R_S ∉ NBIP2_A then cancel
29     decide(UID, R_S, s_S, a, n, v)
```

overhead, we run consensus sessions for individual slots rather than the entire Q_2 content. Each session comprises multiple *attempts*, and each attempt comprises one or more *rounds*. We add the notion of attempt, because we may need to run another attempt of an already decided consensus session, due to the nature of our environment. Users initiate consensus with initial values (v_I) equal to their $Q_1(R_S, s_S)$ content (lines 11–12). If user A has no such content, its initial contribution will be a 'Noop' (or *null*). Any non-'Noop' contribution will be sent for round 1, containing the value (lines 13–20). The $CONTRIBUTION$ message identifies the region, which will enable the subscribers of the region to participate in the consensus. Most consensus algorithms (including OTR), depend on knowing the consensus population (n_S) a priori. We enable a bootstrapping mechanism based on reachability beaconing (similar to [51]), for a user to get an estimate of the population; the number of users eligible to participate for R_S, are the number of total subscribers of R_S and its ancestor nodes in accordance with the namespace hierarchy, *e.g.*, Fig. 1(b). In a highly fragmented network environment that we consider, there is a chance this estimation will be incorrect. To remedy this, we allow the user to update its estimation of n_S, from the contributions it receives, to have an upper bound estimate of n_S.

It is important that users synchronize to be in the same attempt and round as much as possible. Upon receiving a contribution (lines 21–45), the user jumps to the attempt and round number of the contribution message if it is larger than its own (lines 24–28, 36–40). This helps users use the good period fuller when it occurs. Users remove obsolete contributions from the buffer, which helps with scalability and reduces the number of messages circulating in the network. Received contributions from older attempts and rounds will be discarded, with a possible response providing the decision that was already made. When the contribution is in the same attempt and round that the user is in (line 40), the user adds it to its contribution list (line 41). When user's received contribution set reaches the cardinality equal to $2/3 \times n_S$ (one-third rule, line 43), the user will either: 1) start a new round, sending a contribution with the value equal to the smallest (*i.e.*, earliest in terms of causality) most frequently received value (line 44), or 2) decided on a value, if all values in the set are the same (line 45). The decision procedure is described in Alg. 4. A decision message will be published as a result of reaching a decision (line 10). The value in the decision message, determines what value (update) should be inserted into everyone's Q_2 in that particular region's slot (lines 24–26). Another way of reaching a decision is to receive a decision message from someone who has already decided (lines 27–29). CoNICE satisfies the following properties:

Property 2: Consensus. Every consensus session for a $<R, slot>$ pair in the Strong Views preserves the following: 1) **C1:** *Termination*. Every *correct* user (*i.e.*, that does not permanently crash or become unreachable) eventually decides. 2) **C2:** *Agreement*. No two users decide differently. 3) **C3:** *Validity*. Any value decided is some user's initial value. 4) **C4:** *Update Validity*. Any value decided is a valid update that was created. 5) **C5:** *Integrity*. No user decides twice.

Property 3: Total and Causal Order of the Strong View. If users A and B apply region R-bound updates u and u' to their strong views, the order of u and u' is the same in both A and B. This order also respect causality.

CoNICE also supports remedies in additional cases. These cases include loss of causality, duplicate decisions, and long-term fragmentation. The first two cases are elaborated in §V of [47], and the third is discussed in §V.

V. Proof of Consensus in CoNICE

This section provides proof for CoNICE's consensus protocol against its properties described in §IV, using the formalism of Heard-of model [13]. Since Property 1 is about basic causal ordering and not directly on consensus, we presented it in §IV-B instead. This section only focuses on consensus-related properties. Note that *agreement* is the goal, and *consensus* is the procedure to provide it. The major novelty of our proof is extending the earlier proof methods for asynchronous consensus algorithms [13] to cases by also including long-term fragmentation among the users participating in the consensus.

A. Heard-Of Model

We use the *Heard-Of (HO)* model [13] for our proof formalism, as it allows for both node and link failures in asynchronous environments, which is appropriate for our system model. The HO model evolves from one communication-closed round to the next, where messages sent in each round are received and used in the same round. A *Heard-Of set*, denoted by $HO(\pi, r)$, represents the set of users (or *processes*) from whom user π "hears of" (*i.e.*, receives messages that were originated from) in round r. *Communication predicates* specify the features and requirements of a system under the HO model.

A consensus problem is solved in the HO model by a *Heard-Of machine* defined as $M = (\Lambda, P)$, where Λ is an algorithm and P is a communication predicate. Work in [12] uses the HO model to prove the correctness of classic OTR in DTN environments; we extend that to also address invalidations and long-term fragmentation in CoNICE.

The procedure to prove a consensus protocol in the HO model is as follows: We first start by defining the system model and basic assumptions in the environment. We then formally specify communication predicates for network conditions that are necessary and/or sufficient for the specific protocol to work. Using various deductive or inductive reasoning methods, we prove the theorem of the form "under the given predicate, the given protocol solves consensus".

B. System Model

Each run (*i.e.*, attempt) of our consensus protocol pertains to a particular region R_S and a slot number s_S (for a consensus session S). The slot number for an update is the next available empty slot in the queue associated with R_S across different users. Its participants are the subscribers of region R_S, in accordance with the namespace. An important novelty of CoNICE is that it can solve consensus even in the case of long-term fragmentation, within the same run of the algorithm (§IV-C). Thus, we define two types of consensus solutions for each user within the same run: 1) *local consensus* (to reach *local agreement*), and 2) *global consensus* (to reach *global agreement*). Users constrained within the same long-term fragment (*e.g.*, trapped in a shelter during disaster) can reach local agreement amongst themselves for a $<R_S, s_S>$ pair through the one-third rule. Upon re-connecting of the shelter to the rest of the network after some time, if there are conflicts between the different values decided based on local agreements and as a result of invalidations of all but one of the values (as described in §IV-C), users can reach global agreement throughout the whole network.

The basic assumptions of our system model are as follows:

- We assume each user in the network is equipped with the CoNICE protocol stack and follows the protocols honestly and correctly (performs reachability beaconing, counts group population per region, *etc.*).
- In a network without long-term fragments, proving our consensus is similar to that of [12]. Thus, we only focus on the fragmented scenarios here.
- We assume that we have multiple disjoint long-term fragments where each fragment Δ_i includes a set of users Π_i with n_i members who are subscribers of R_S. Π is the set of all subscribers of R_S throughout the whole network.
- No more than $\frac{n_i}{3}$ of users permanently crash. Also, in each round, at least one user hears from at least $\frac{2}{3}$ of the other users in the fragment:

$$\forall r,\, \exists \pi \in \Pi_i \, s.t. \, |HO(\pi, r)| \geq 2n_i/3 \tag{1}$$

  This is reasonable to assume for our environment, as we use the gossiping protocol that leverages mobility and buffering at users, and mules, to deliver messages with a high delivery probability, especially within a shelter. Furthermore, CoNICE's reachability beaconing and exchange of population counts makes sure that each user has a good enough approximation of n_i, which is needed for the OTR-based calculations.
- After a significant period of time, paths (and connections) appear among some of the fragments,

We first focus on solving local consensus within a fragment (*i.e.*, before there appears any paths between fragments). In that fragment, there is a set of participants Π_i running CoNICE's OTR-based algorithm for $<R_S, s_S>$:

$$\Pi_i = \{\pi_i | \pi_i \in \Delta_i \wedge \pi_i \in sub(R_S)\} \tag{2}$$

We specify the following communication predicate P_{otr}:

$$P_{otr} : \forall r > 0, \exists r_0 \geq r : \exists \Pi_0 \in \Pi_i \, s.t. \, |\Pi_0| \geq 2n_i/3 \\ \wedge \forall \pi \in \Pi_i : HO(\pi, r_0) = \Pi_0 \tag{3}$$

which says that infinitely often, there will be rounds in which all the users will hear from a two-third majority subset of members, which is Π_0. If P_{otr} holds, CoNICE solves consensus since after several rounds, all users will deterministically converge their contribution to the same value (lines 40–45 in Alg. 3) and eventually agree on a value and decide. Similar to what is discussed in [12], predicate P_{otr} is a sufficient condition to solve our consensus. Here, we complement that specification, *i.e.*, P_{otr}, by specifying a simpler predicate P_{L1} which is the necessary condition to achieve agreement:

$$P_{L1} : \exists r_0, v \, s.t. \, \forall \pi_i \in \Pi_i : |HO(\pi_i, r_0)| \geq 2n_i/3 \wedge v_i^{r_0} = v \tag{4}$$

where $v_i^{r_0}$ is the value picked for either contribution or decision at the end of round r_0 at user π_i. Predicate P_{L1} says that in at least one round, all users will pick the same value, and they hear it from at least $\frac{2}{3}$ of the users, even if it is not the same $\frac{2}{3}$ subset across everyone. If v is the smallest most frequent non-'Noop' value from received contributions, but not $\overline{V}$ (as defined in line 45), another predicate is needed (P_{L2}) so that in the immediate next round, all non-crashed users could exchange their contributions & hear from a $\frac{2}{3}$ set:

$$P_{L2} : \exists r_1 > r_0 \, s.t. \, \forall \pi_i \in \Pi_i : |HO(\pi_i, r_1)| \geq 2n_i/3 \tag{5}$$

In §V-C, we will show that with $P_L = P_{L1} \wedge P_{L2}$, CoNICE solves consensus in this case. Alternately, if v is the received decision value (lines 2–11 in Alg. 4), CoNICE will also solve consensus (more details in §V-C).

To investigate global consensus, let us now assume that after a significant period of time, paths appear among some of the fragments, some of those Δ_i's have potentially reached their local decisions at round r_i^d on their respective value v_i^d. To follow the model's round-based specification, we define a special round r_∞, which involves every message exchange after the re-connection of all these fragments (which have reached local agreement), for $<R_S, s_S>$ for the rest of the lifetime of the whole network. All the invalidation procedures of decisions previously reached within the same attempt (line 13–20 in Alg. 4) occur in r_∞. With respect to r_i^d and r_∞, each user in Δ_i can be in one of the following four categories,

which determines their status of whether or not they can decide for local and/or global consensus:

1) user enters r_i^d, and then enters r_∞.
2) user never enters r_i^d, but enters r_∞.
3) user enters r_i^d, but never enters r_∞.
4) user neither enters r_i^d nor r_∞.

CoNICE's global agreement only involves users in categories 1 and 2, *i.e.*, those that enter r_∞, as interaction and contention between different decisions from different fragments is needed, *i.e.*, to further resolve to reach agreement. Users in categories 3 and 4 are permanently isolated in their fragments, and thus not relevant for global consensus, with the relative advantage of users in category 3 over those in category 4 being that the achievement of local consensus can be locally useful, *e.g.*, within a shelter, in category 3.

Among the different fragments that have reached local agreement and entered r_∞, let us define Δ_{max} to be the fragment with the highest population, namely n_{max}, *i.e.*, the "winning" fragment (line 15–16 in Alg. 4). The decided value for Δ_{max}'s local consensus is denoted by v_{max}^d. If there is more than one fragment with the highest population, *i.e.*, "tie" situations (lines 17–19 in Alg. 4), Δ_{max} is the fragment with the highest population *and* includes the non-crashed user with highest UID, denoted by U_{max}. We define these predicates:

$$P_{G1} : \forall \Delta_j \neq \Delta_{max} : n_j < n_{max} \implies \forall \pi \in \Pi : \exists \pi' \in \Delta_{max}\, s.t.\, \pi' \in HO(\pi, r_\infty) \quad (6)$$

$$P_{G2} : \exists \Delta_j \neq \Delta_{max}\, s.t.\, n_j = n_{max} \implies \forall \pi \in \Pi : U_{max} \in HO(\pi, r_\infty) \quad (7)$$

Predicate P_{G1} says that if Δ_{max} is the uniquely most populated fragment, every other user will hear from some user in Δ_{max} in r_∞. Predicate P_{G2} says that if population of Δ_{max} is tied, all users hear from U_{max} in r_∞. It is worth noting that P_{G2} does not necessitate the permanent existence of the user U_{max} in the network; as long as U_{max} stays in the round r_∞ long enough to send its local decided value to some other user, the gossiping protocol enables that message to reach others through D2D relaying. As per Alg. 4, users who receive a message, first check the population count, and then the user ID carried in the decision message, to converge towards the correct global decision value. Also, no additional majority check or additional rounds are needed in the global agreement phase (due to pair-wise comparison in lines 14–20 in Alg. 4). Thus, the communication predicate built by the conjunction of the two, namely $P_G = P_{G1} \wedge P_{G2}$, is necessary and sufficient for CoNICE to ensure global agreement, which we show in more detail in §V-C.

C. Proving Property 2: Basic Consensus Property

We now investigate if CoNICE's consensus protocol, in particular Alg. 3 and 4, satisfy the consensus property 2, and its sub-properties C1–5, under the communication predicates explained in §V-B.

C1: Termination. For local consensus, assuming P_{L1} stands: 1) if all values v are mere contributions, then Alg. 3 ensures that each user reaches the decision mode in the next round (assuming P_{L2}), decides and terminates; 2) if at least one of those v's is a decision message, Alg. 4 (for the same and different attempt, respectively) ensures that the user jumps to the decision mode and terminates. For global consensus, if P_G stands and users do not permanently crash, Alg. 4 ensures that CoNICE always favors the decision received from the winning fragment/user; thus global consensus will eventually terminate since P_G ensures that the winning decision will be eventually heard of in r_∞ by all users.

C2: Agreement. Local agreement specifies that after some round r_i^d at Δ_i, the value for $<R_S, s_S>$ at each user in Δ_i (*i.e.*, set of users Π_i), will be the same, namely $v_i^d = v$. We assume P_L holds at some round r_0. We aim to prove local agreement is preserved at any round r after r_0. We prove that proposition using induction on the value of $r{-}r0$:

Basis: $r = r_0 + 1$. In this case, all users have picked value v at r_0; if v is the next contribution value that all users pick, then in the next immediate round $(r_0 + 1)$, as long as a user hears of "any" $\frac{2}{3}$ set of users, the conditions in both lines 43 & 45 in Alg. 3 will be true. Thus, it ensures all users go to decision, with value v; so at round r, all users agree on v.

Inductive step: Suppose that at $r = r_0{+}n$, all users agree on value v. In round $r = r_0{+}n{+}1$, we assume users hear of each other. Given that users receive a duplicate decision value v in the same attempt, their decision does not change (line 13 in Alg. 4). If users receive v as a contribution message, given that they already made a decision, then the decision status does not change, ensured by lines 30-32 in Alg. 3. As a result, in both cases, the decision value v within the same attempt remains unchanged, which proves the induction for $r = r_0 + n + 1$, proving CoNICE's local agreement.

For global consensus, we assume predicate P_G holds. As C1 shows, every user in the network terminates with the correct final value v, which is v_{max}^d, albeit possibly going through the incremental changes, based on the "highest winner seen so far" by a user (Alg. 4). Since all users converge to termination on the same value (shown by C1), eventually every user in Π will agree on v_{max}^d. Any user that has not yet reached v_{max}^d is a user that has not yet terminated. Similarly, one that crashes permanently is not considered in the agreement requirement. This proves CoNICE's global agreement property. Therefore, any consensus run according to CoNICE's protocol, *i.e.*, all consensus steps that do not require a new attempt or a whole new session, preserves agreement.

C3: Validity. Alg. 3 ensures that every user starts a consensus session with an initial value. If it is not 'Noop', line 12 ensures that the initial value is then used to create contributions that are disseminated. As we can see in Alg. 3, no new value is created during the message exchange among users; thus the final value picked for a decision is necessarily some user's initial value, guaranteeing the validity property.

C4: Update Validity. For update validity, every user starts with an initial value that he has delivered in level 1, and thus is a valid update. It can also be a 'Noop'. Alg. 3 makes sure that 'Noop' would never pass the one-third rule test and proceed to the decision. With this fact and also considering C3, CoNICE ensures that update validity holds.

C5: Integrity. When there is no decision invalidation, C1 ensures integrity is also guaranteed, as no user will make

a duplicate decision. In the case of invalidation, this property still holds, since at any point in time, only one decision is valid, even for the case of multiple attempts for a consensus session. This is ensured by line 9 in Alg. 3, as the variable dec_S by a user, only holds at most one value at a time, which is its latest valid value.

D. Proving Property 3: Total and Causal Ordering Property

In Property 2, we proved the consensus properties of CoNICE, regrading individual consensus sessions. Property 3 has two separate parts, both of which we will consider. The first part of property 3 specifies total order. Suppose two users π_1 and π_2 have established their final sequences of updates for region R in their strong views, without any holes in the sequence. By way of proof by contradiction, let us also suppose that π_1 has placed updates u and u' in slots s_1 and s_2 respectively (with $s_1 < s_2$), and user π_2 has placed updates u' and u in slots s_3 and s_4 respectively (with $s_3 < s_4$). In other words, we are supposing π_1 placed u before u', while π_2 placed u' before u, thus violating total ordering. We consider the following possible cases:

1. No invalidations are needed: This case would entail that π_1 decided u for $<R, s_1>$ while π_2 decided some update $u'' \neq u$ for $<R, s_1>$ (with similar situation for u'). This would mean that the consensus protocol reached two different decisions u and u'' for the same session $<R, s_1>$, violating the agreement property. Since there is no need for invalidation and CoNICE ensures agreement, this case is impossible.

2. There has been loss of causality, thus invoking the re-ordering procedure (invalidation case 1 in §IV-C): This case would entail that update u was initially inserted into some slot s_5 at both π_1 and π_2 (with similar situation for u'), but the re-ordering due to loss of causality changed that. This would mean that the local re-ordering step (lines 25–26 in Alg. 4) performed two different ordering outcomes at π_1 and π_2. As the re-ordering module used in Alg. 4 is deterministic (*i.e.*, same input leads to same output), this is impossible.

3. There has been long-term fragmentation, thus invoking the contention-breaking procedure for contention between fragments (invalidation case 2 in §IV-C): This would mean that the consensus protocol reached two different decisions u and u'' for the same session $<R, s_1>$, after the re-connection of fragments, violating the agreement property. Since the invalidation procedure for long-term fragmentation picks one single value to eventually converge to (*i.e.*, value from the winning fragment/user), and CoNICE ensures agreement even in case of global consensus, this case is impossible.

4. There has been duplicate decisions and thus a new attempt procedure is invoked (invalidation case 3 in §IV-C): Similar to the previous case, this would require that the consensus session for $<R, s_1>$ reach two different values u and u'' (with similar situation for u'), after a new attempt. Given that CoNICE ensures agreement in each of its attempts (no matter how many re-attempts happen), this is impossible.

Therefore, the initial hypothesis about lack of total order is contradicted, and CoNICE's total ordering is proved. As for causal order, since the outcome of the decided sequences get deterministically sorted (by the aforementioned sorting method in case 2 above) and re-ordered for causal order if necessary, causal order of the Strong view is also guaranteed.

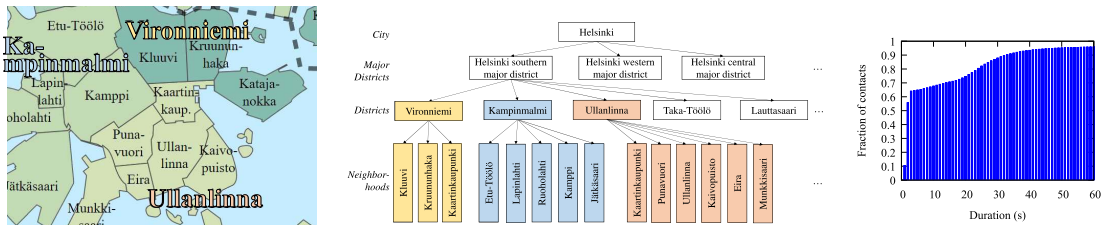
VI. Evaluation

To evaluate CoNICE, we perform a simulation based on a partial map of the city of Helsinki (Fig. 3(a), [52], with more details in [23]) using the ONE simulator [53]. The associated hierarchically-structured namespace (Fig. 3(b)) follows the "City→Major districts→Districts→Neighborhoods" structure. Our simulation environment consists of the three districts (and hierarchically, the neighborhoods in them) highlighted in Fig. 3(a), and is 4500×3400 meters large. Our simulation setup mimics the way first responders get mobilized and move around neighborhoods for a search and rescue operation in natural disasters, such as a hurricane. We model an emergency response scenario where there are 30 pedestrian first responder users (F-users), each dealing with one of the three districts: they are moving in the area, indicate an interest in events in them, and publishing updates for them. There is no networking infrastructure, but all users are equipped with D2D wireless capability. To increase message delivery, we place additional benevolent mules, namely 500 pedestrian civilians (C-users) and patrol vehicles (V-users). V-users move faster, have higher buffer capacity and wireless range than pedestrian users. Benevolent mules participate in relaying and causal delivery of every message they receive (regardless of region). However, they do not participate in any consensus sessions. We assume all users in these scenarios, *i.e.*, F-users, C-users, and V-users, behave honestly and non-maliciously. Mobility is based on map routes, with waiting times of at most 2 min. Each F-user creates three updates in the first half hour of the simulation (thus, total of 90 uniquely created updates), randomly belonging to one of the neighborhoods in their respective district. All messages are 1 KB. We report on two sets of scenarios, one with 1 hour in simulated time and another for 12 hours.

A. Experiments on Gossiping and Causal Ordering

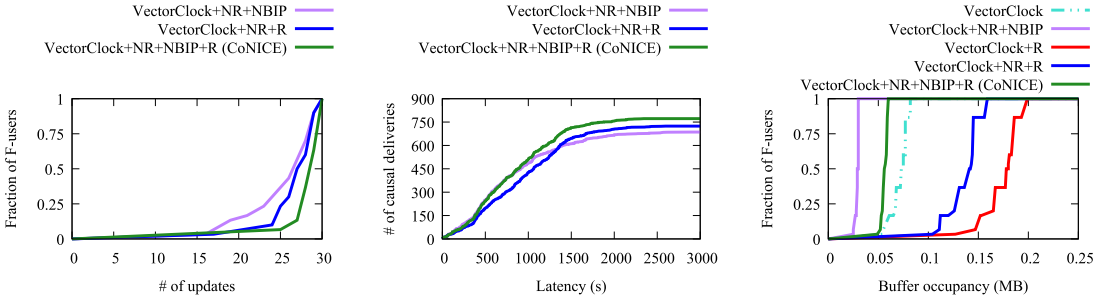
To investigate level 0 and level 1 consistency, we use the 1-hour simulation scenario. There are a total of 59,558 D2D contacts during this time, and the cumulative distribution of contact durations is (partially) shown in Fig. 3(c). As Fig. 3(c) shows, 95 percent of contacts lasted less than 1 minute and 70 percent less than 10 seconds, which demonstrates the highly dynamic nature of the environment. The mobility and contact distribution is the same for all experiments in this sub-section.

First, we focus on gossiping only (*i.e.*, no causal ordering or consensus). We define *replication coverage* (RC) as a metric that shows how many of updates each node has received (albeit out of order). *Total* RC (RC_{tot}) denotes all updates a user received, while *relevant* RC (RC_{rel}) only considers the relevant ones pertaining to the F-user's tasks (can be at most 30). Note that always $RC_{rel} \leq RC_{tot}$, and with the right interest profiling, it is expected that $RC_{rel} = RC_{tot}$. CoNICE's gossiping enhances epidemic routing. As Table I shows, CoNICE achieves better RC_{rel} than 'epidemic routing+NR' (NR is name-based region-ing for publications), as it adds name-based interest profiling (NBIP). It also achieves better



(a) Map of Helsinki simulation scenario (b) Namespace for our Helsinki-based simulation scenario (c) Contact duration CDF

Fig. 3. Simulation scenario.



(a) Relevant causal completeness CDF (b) Relevant causal delivery (cumulative) (c) Buffer occupancy with levels 0 and 1

Fig. 4. Results for level 1 (causal ordering).

TABLE I

RESULTS FOR LEVEL 0

Metric/Approach		Epidemic Routing	Epidemic Routing+NR	EpidemicRouting+ NR+NBIP (CoNICE)
F-users	Average RC_{rel}	N/A	28.40	29.53
	Average RC_{tot}	73.60	74.76	29.53
	Average RL_{rel} (s)	N/A	852.25	758.89
	Average RL_{tot} (s)	1,080.07	1,084.17	758.89
	Average Buffer Occupancy (MB)	0.07	0.07	0.02
Net-work	Total Relays	49,612	50,123	48,612
	Irrelevant Relays	N/A	1,393	0

TABLE II

RESULTS FOR LEVEL 1

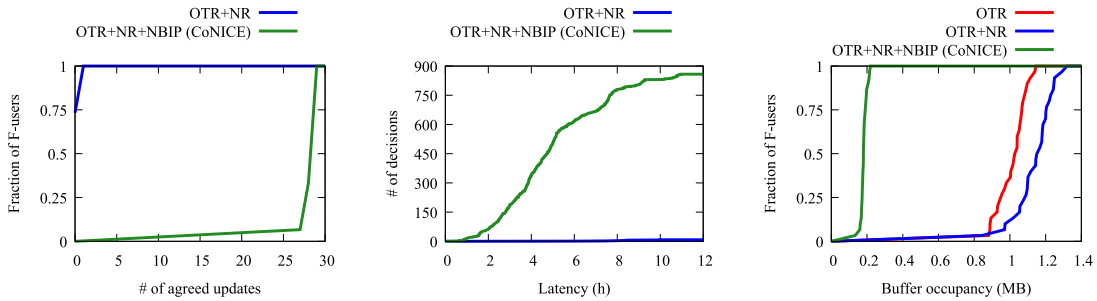
Metric/Approach		Vector Clock	Vector Clock+ NR+NBIP	Vector Clock +R	Vector Clock+ NR+R	VectorClock+ NR+NBIP+ R (CoNICE)
F-users	Average CC_{rel}	N/A	25.73	N/A	28.30	28.70
	Average CC_{tot}	39.86	25.73	68.30	71.16	28.70
	Average CL_{rel} (s)	N/A	693.23	1088.16	800.18	729.31
	Average CL_{tot} (s)	1,093.01	693.23	1,119.02	1,084.79	729.31
	Average Buffer Occupancy (MB)	0.07	0.02	0.17	0.13	0.05
Net-work	Total Relays	49,612	98,485	108,289	88,134	89,792
	Irrelevant Relays	N/A	0	N/A	2,648	0

latency with more relevant deliveries (RL_{rel}), (at most can reach 30×30.900). This is due to naming which makes relays and queued messages more useful and relevant. Also, basic epidemic routing that uses no naming (the notion of 'relevancy' not applicable), receives lower RC_{tot} than when enhanced with NR. Its total relays value is similar to the rest while achieving less. CoNICE achieves higher coverage with lower buffer and network cost. More results are in [47].

We then bring causal ordering into play. CoNICE's causal ordering enhances Vector Clock with the use of NR, NBIP, and Reactive mode (R), in addition to other minor optimizations such as variable-length vectors. Table II provides a comparison summary. Fig. 4(a) shows the CDF of relevant causal completeness (CC_{rel}), which denotes how many of updates have been causally applied at F-users. As Fig. 4(a) shows, CoNICE achieves better CC_{rel} compared to alternatives that enable NR, since CoNICE allows more selective use of causal ordering overhead (through NBIP) and requesting for unfulfilled prerequisites on demand using the reactive mode rather than waiting. It also achieves better causal latency (CL_{rel}) as Fig. 4(b) shows, and reasonable network overhead in terms of the number of relays (Table II). For cases *without NR*, Table II shows that pure Vector Clock achieves the lowest RC_{tot}. This is because without name-based region-ing, every update can potentially depend on all others, which results in an extremely high number of references that have to be fulfilled and processed. Name-based region-ing makes it more selective, having to only depend on relevant updates. Fig. 4(c) shows CoNICE achieves better buffer usage than most alternatives, except 'VC+NR+NBIP' which does not use reactive mode and achieves lower completeness. As seen, using causal ordering leads to slightly higher latency and buffer usage than pure level 0, but achieves causal order consistency.

B. Experiments on Consensus

To investigate consensus, we extend our scenario to 12 hours with the total of 683,876 D2D contacts. We now enable level 2, *i.e.*, consensus, on top of levels 0 and 1. After the passage of approximately one hour, users start to initiate consensus sessions for the slots they have content for. We compare CoNICE with the basic OTR, and show the impact of adding NR and NBIP. The *agreement completeness* (AC) metric shows how many of level 2 queue slots of users have been filled with agreed-upon updates. Just as before, we have AC_{rel} and AC_{tot}. As Table III shows, basic OTR fails to reach any decisions, and thus has zero AC. 'OTR+NR' is slightly better but is still not satisfactory. As the Table, and Fig. 5(a) (CDF of AC_{rel}) show, CoNICE achieves a dramatically better



(a) Relevant agreement completeness CDF (b) Relevant decision latency (cumulative) (c) Buffer occupancy with levels 0–2

Fig. 5. Results for level 2 (consensus).

TABLE III
RESULTS FOR LEVEL 2

Metric/Approach		OTR	OTR+NR	OTR+NR+NBIP (CoNICE)
F-users	Average AC_{rel}	N/A	0.26	28.60
	Average AC_{tot}	0	0.93	28.60
	Average AL_{rel} (h)	N/A	8.29	4.91
	Average AL_{tot} (h)	None	7.51	4.91
	Average Buffer Occupancy (MB)	1.01	1.14	0.18
Network	Total Relays	3,489,035	3,512,598	3,504,557
	Irrelevant Relays	N/A	77,086	0
	Consensus Initiations	2,049	2,101	853
	Consensus Decisions	0	28	858

TABLE IV
ENERGY CONSUMPTION: WITH SUFFICIENT INITIAL ENERGY

		OTR+NR		OTR+NR+NBIP (CoNICE)	
E_{scan}	$E_{transmit}$	Avg. AC_{rel}	Avg. NEC	Avg. AC_{rel}	Avg. NEC
1	10	0.26/30	86,531.37	28.6/30	72,493.37
1	1	0.26/30	47,796.20	28.6/30	46,445.47
10	1	0.26/30	439,102.37	28.6/30	437,588.60

agreement completeness. Fig. 5(a) shows that with CoNICE, 90% of F-users agree on 26 or more updates, while with 'OTR+NR', 75% of F-users agree on zero updates, in the entire 12-hour simulation period. This is due to the fact that CoNICE uses NBIP, which limits the consensus participants only to those that are relevant, namely F-users dealing with neighborhoods within the same district. Table III also shows that OTR and 'OTR+NR' initiate much higher consensus sessions than CoNICE (2,049 and 2,101 *vs.* 853), but reach significantly fewer decisions (0 and 28 *vs.* 858). CoNICE even reaches more decisions than it initiates, which shows the improvement contributed by level 2 over level 1. This is because due to CoNICE's faster consensus convergence, some F-users can fill their slots in Q_2 the corresponding of which they do not have in Q_1. Fig. 5(b) shows the cumulative latency of reaching relevant agreement decisions (AL_{rel}) across all F-users. As shown, CoNICE achieves considerably more. As can be seen (and previously shown in [12]), the latency of reaching consensus decisions is on the scale of hours in an intermittently-connected network, while CoNICE's causal order delivery is in the order of minutes (Table II). This shows yet another benefit of going through level 1 first and then level 2: users will have a somewhat useful moderate view in the order of minutes while dealing with the incident, while waiting for possibly hours to reach consensus and build a strong view. CoNICE achieves far better agreement completeness, using the same level of relays as other alternatives (Table III), and using much less buffer at F-users as shown in Fig. 5(c). These results show that CoNICE significantly improves on OTR, for achieving higher agreement completeness among users, while also using less buffer capacity. These improvements of CoNICE are greatly beneficial in practical situations such as geo-tagging in emergency response, as first responders can build their consistent strong views much faster and be able to deal with their critical tasks more effectively and efficiently.

C. Energy Consumption

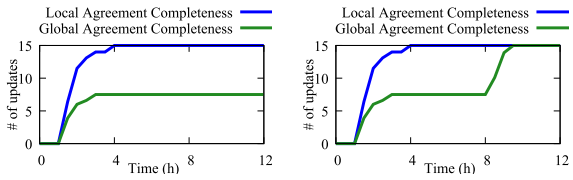
An important aspect in our application is energy consumption. Using our simulation environment, we investigate a first responder's mobile device energy consumption resulting from network-related operations: wireless scanning and data transmission. We compare the use of CoNICE's NBIP *vs.* without NBIP. We consider two cases with regard to the initial energy, whether it is sufficient or insufficient.

In the first case, we assume all user devices have sufficient (battery) energy to begin with, so that they do not run out of battery power during a 12-hour simulation duration. Table IV shows the *network energy consumption* (NEC) as well as *relevant agreement completeness* (AC_{rel}), both averaged among all F-users (first responders). We compare for different values (and ratios) of *scan energy* (E_{scan}) and *transmission energy* ($E_{transmit}$). E_{scan} is the amount of energy used for a single request or response message in the process of scanning/discovery a neighboring device. $E_{transmit}$ is the amount of energy used when transmitting a byte of data per second. All energy-related values in the Table (*i.e.*, E_{scan}, $E_{transmit}$, and NEC) are measured in "simulated energy units" (*e.g.*, KJoules). As Table IV shows, CoNICE achieves much better agreement completeness with less energy consumption. This is due to fewer data transfers needed, thanks to the selectivity brought about by NBIPs. This energy reduction is more apparent when the $E_{transmit}$ to E_{scan} ratio is higher. The scanning activity is the same regardless of the consensus protocol, as it only depends on the mobility pattern.

In the second case, we focus on the *initial energy*. Limiting it may cause insufficient energy for consensus sessions to continue until the end of 12^{th} hour. We consider different initial energy values, with values of E_{scan} and $E_{transmit}$ set to 1. Table V compares the use of NBIP in CoNICE *vs.* not using it. The Table shows that while $50kJ$ is sufficient

TABLE V
ENERGY CONSUMPTION: INSUFFICIENT INITIAL ENERGY

			OTR+NR		OTR+NR+NBIP (CoNICE)	
Initial Energy	E_{scan}	$E_{transmit}$	Avg. AC_{rel}	Avg. NEC	Avg. AC_{rel}	Avg. NEC
40k	1	1	0/30	40,000.00	28.3/30	40,000.00
50k	1	1	0.26/30	47,796.20	28.6/30	46,445.47



(a) Basic OTR (no invalidations) (b) CoNICE (OTR + invalidations)

Fig. 6. Average local and global agreement completeness.

initial energy for both approaches, reducing it to $40kJ$ causes both approaches to not reach their full potential: 'OTR+NR' will result in zero completeness, while 'OTR+NR+NBIP' (CoNICE) achieves the completeness ratio of 28.3 out of 30. As these results show, CoNICE achieves much better relevant agreement completeness, even when running out of battery energy during the consensus procedure.

D. Physical Fragmentation of Shelters During Disasters

We now conduct experiments to investigate the effect of long-term fragmentation: we have two long-term fragmented shelters each circular with radius of 50 meters, with 500 meters distance between them. Each shelter contains 5 first responders (F-users), all dealing with the disaster in the 'Vironniemi' district. These F-users are constrained (or trapped) with very limited mobility within their respected shelters. Every F-user creates 3 updates. Thus, we will have a total of 30 updates. All benevolent mule activity is disabled during the first 8 hours of the experiment, with no connection between members of the two shelters. Due to this long-term fragmentation, the two shelters do not hear from each other for the 8-hour period, and we end up with 15 pairwise contentious consensus sessions, *i.e.*, same region-slot pair, from the two shelters.

Fig. 6 compares the use of basic OTR (without invalidation), (a), and CoNICE (OTR with invalidation), (b), for achieving both local and global agreement (as defined in §V). Local agreement completeness indicates to what extent an average F-user reaches the same decision for an update as the others in the same shelter. Global agreement completeness, on the other hand, represents to what extent an average F-user reaches the "winning" global decision for an update just like the others in the whole network. Both are shown as per-user average, and each can be at most 15.

As Fig. 6(a) shows for the basic OTR approach, it is easy for all the F-users to gradually reach complete local agreement before the 8th hour. Since one shelter has the winning advantage, at most half global agreement is reached (*i.e.*, those who are already in the winning shelter), without any invalidation procedure. We also observe the same outcome if we restart the whole consensus procedure with all 10 participants. This occurs because there is insufficient time to reach global agreement among all users. Thus, we do not make progress in achieving global agreement even after 12 hours. Alternatively, as Fig. 6(b) shows, with the addition of decision invalidation procedure, CoNICE achieves complete global consensus among all F-users in the network, gradually starting from the 8th hour, which is the time paths start to appear between shelters. This feature of CoNICE helps ensure that all first responders can get a chance to eventually get on the same page regarding the ordering of updates for their respective tasks, in a strongly consistent manner, even if they have been isolated in separate shelters for a long time.

VII. CONCLUSION

We proposed CoNICE, a framework to ensure consistent dissemination of updates among users in intermittently-connected environments. It exploits naming and multi-level consistency for more selective and efficient causal ordering and consensus. We proved CoNICE guarantees consensus with causal and total ordering properties. Additionally, we showed that CoNICE can also solve consensus even in case of long-term fragmentation. Our simulation experiments on an application of map-based geo-tagging in emergency response show that CoNICE achieves a considerably higher degree of agreement completeness than the state-of-the-art asynchronous consensus algorithm, OTR, as it exploits naming, showing the applicability of CoNICE in practical, intermittently-connected scenarios.

REFERENCES

[1] M. Jahanian, Y. Xing, J. Chen, K. K. Ramakrishnan, H. Seferoglu, and M. Yuksel, "The evolving nature of disaster management in the internet and social media era," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Jun. 2018, pp. 79–84.
[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2003, pp. 27–34.
[3] C. Boldrini, K. Lee, M. Önen, J. Ott, and E. Pagani, "Opportunistic networks," *Comput. Commun.*, no. 48, pp. 1–4, Oct. 2014.
[4] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in LTE-advanced networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1923–1940, 4th Quart., 2015.
[5] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable Secure Distrib. Programming*, 2nd ed. Berlin, Germany: Springer, 2011.
[6] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.
[7] D. R. Cheriton and D. Skeen, "Understanding the limitations of causally and totally ordered communication," in *Proc. 14th ACM Symp. Oper. Syst. Princ.*, 1993, pp. 44–57.
[8] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.
[9] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[10] M. Swan, *Blockchain: Blueprint for a New Economy*. Newton, MA, USA: O'Reilly Media, 2015.
[11] F. Borran, R. Prakash, and A. Schiper, "Extending paxos/lastvoting with an adequate communication layer for wireless ad hoc networks," in *Proc. Symp. Reliable Distrib. Syst.*, Oct. 2008, pp. 227–236.
[12] A. Benchi, P. Launay, and F. Guidec, "Solving consensus in opportunistic networks," in *Proc. Int. Conf. Distrib. Comput. Netw.*, Jan. 2015, pp. 1–10.
[13] B. Charron-Bost and A. Schiper, "The heard-of model: Computing in distributed systems with benign faults," *Distrib. Comput.*, vol. 22, no. 1, pp. 49–71, Apr. 2009.
[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol.*, 2009, pp. 1–12.
[15] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.

[16] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan, "COPSS: An efficient content oriented publish/subscribe system," in *Proc. ACM/IEEE 7th Symp. Archit. Netw. Commun. Syst.*, Oct. 2011, pp. 99–110.

[17] I. Psaras, L. Saino, M. Arumaithurai, K. K. Ramakrishnan, and G. Pavlou, "Name-based replication priorities in disaster cases," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Apr. 2014, pp. 434–439.

[18] S. S. Adhatarao, J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "Comparison of naming schema in ICN," in *Proc. IEEE Int. Symp. local Metrop. Area Netw. (LANMAN)*, Jun. 2016, pp. 1–6.

[19] A. Venkataramani, J. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "MobilityFirst: A mobility-centric and trustworthy internet architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 74–80, Jul. 2014.

[20] M. Jahanian, J. Chen, and K. K. Ramakrishnan, "Graph-based namespaces and load sharing for efficient information dissemination in disasters," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–12.

[21] J. R. G. Paz, "Introduction to azure cosmos db," in *Microsoft Azure Cosmos DB Revealed*. Berkeley, CA, USA: Apress, 2018, pp. 1–23.

[22] F. Houshmand and M. Lesani, "Hamsaz: Replication coordination analysis and synthesis," *Proc. ACM Program. Lang.*, vol. 3, pp. 1–32, Jan. 2019.

[23] *CoNICE: Consensus in Name-Based Intermittently-Connected Environments*. Accessed: Jan. 1, 2022. [Online]. Available: https://github.com/mjaha/CoNICE

[24] K. K. Ahmed, M. H. Omar, and S. Hassan, "Survey and comparison of operating concept for routing protocols in DTN," *J. Comput. Sci.*, vol. 12, no. 3, pp. 141–152, Mar. 2016.

[25] A. Vahdat *et al.*, "Epidemic routing for partially connected ad hoc networks," Duke Univ., Durham, NC, USA, Tech. Rep. CS-200006, 2000.

[26] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," *IEEE/ACM Trans. Netw.*, vol. 14, no. 3, pp. 479–491, Jun. 2006.

[27] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. ACM SIGCOMM Workshop Delay-Tolerant Netw.*, 2005, pp. 252–259.

[28] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE rap: Social-based forwarding in delay-tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 11, pp. 1576–1589, Nov. 2011.

[29] W. Moreira, P. Mendes, and S. Sargento, "Opportunistic routing based on daily routines," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2012, pp. 1–6.

[30] W. Moreira, P. Mendes, and S. Sargento, "Social-aware opportunistic routing protocol based on user's interactions and interests," in *Proc. Int. Conf. Ad Hoc Netw.*, 2013, pp. 100–115.

[31] H. Lenando and M. Alrfaay, "EpSoc: Social-based epidemic-based routing protocol in opportunistic mobile social network," *Mobile Inf. Syst.*, vol. 2018, Apr. 2018, Art. no. 6462826.

[32] R. Koch, R. Moser, and P. Melliar-Smith, "Global causal ordering with minimal latency," in *Proc. Int. Conf. Parallel Distrib. Comput. Netw.*, 1998, pp. 262–267.

[33] J. Fidge, "TimesTamps in message-passing systems that preserve the partial ordering," in *Proc. 11th Austral. Comput. Sci. Conf.*, 1988, pp. 56–66.

[34] F. Mattern, "Virtual time and global states of distributed systems," in *Proc. Parallel Distrib. Algorithms*, 1989, pp. 215–226.

[35] M. Singhal and A. Kshemkalyani, "An efficient implementation of vector clocks," *Inf. Process. Lett.*, vol. 43, no. 1, pp. 47–52, Aug. 1992.

[36] P. Bailis, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "The potential dangers of causal consistency and an explicit solution," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, pp. 1–7.

[37] R. J. de Araújo Macêdo, "Causal order protocols for group communication," in *Proc. Brazilian Symp. Comput. Netw. (SBRC)*, 1995, pp. 265–283.

[38] E. Gafni, "Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony," in *Proc. 17th Annu. ACM Symp. Princ. Distrib. Comput. (PODC)*, 1998, pp. 143–152.

[39] F. Brasileiro, F. Greve, A. Mostéfaoui, and M. Raynal, "Consensus in one communication step," in *Proc. Int. Conf. Parallel Comput. Technol.*, 2001, pp. 42–50.

[40] M. Jahanian and K. K. Ramakrishnan, "Name space analysis: Verification of named data network data planes," in *Proc. 6th ACM Conf. Inf.-Centric Netw.*, Sep. 2019, pp. 44–54.

[41] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," in *Proc. 2nd ACM Conf. Inf.-Centric Netw.*, 2015, pp. 9–18.

[42] T. Li, Z. Kong, S. Mastorakis, and L. Zhang, "Distributed dataset synchronization in disruptive networks," in *Proc. IEEE 16th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Nov. 2019, pp. 428–437.

[43] D. Tarr, E. Lavoie, A. Meyer, and C. Tschudin, "Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications," in *Proc. 6th ACM Conf. Inf.-Centric Netw.*, Sep. 2019, pp. 1–11.

[44] L. Wang *et al.*, "Naxos: A named data networking consensus protocol," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun.*, 2018, pp. 986–991.

[45] *Urban Search and Rescue—Marking Systems*. Accessed: Jan. 1, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Urban_search_and_rescue#Marking_systems

[46] *Katrina+5: An X-Code Exhibition*. Accessed: Jan. 1, 2022. [Online]. Available: https://southernspaces.org/2010/katrina-5-x-code-exhibition/

[47] M. Jahanian and K. K. Ramakrishnan, "CoNICE: Consensus in intermittently-connected environments by exploiting naming with application to emergency response," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–12.

[48] C. A. Lee, Z. Zhang, Y. Tu, A. Afanasyev, and L. Zhang, "Supporting virtual organizations using attribute-based encryption in named data networking," in *Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC)*, Oct. 2018, pp. 188–196.

[49] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 566–600, 1st Quart., 2018.

[50] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?" *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 41–49, Sep. 2018.

[51] D. Cavin, Y. Sasson, and A. Schiper, "Consensus with unknown participants or fundamental self-organization," in *Int. Conf. Ad-Hoc Netw. Wireless*, 2004, pp. 135–148.

[52] Wikipedia. *Subdivisions of Helsinki*. Accessed: Jan. 1, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Subdivisions_of_Helsinki

[53] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for DTN protocol evaluation," in *Proc. 2nd Int. Conf. Simulation Tools Techn.*, 2009, pp. 1–10.

Mohammad Jahanian received the B.S. degree from the University of Tehran in 2012, the M.S. degree from the Sharif University of Technology in 2014, and the Ph.D. degree from the University of California at Riverside, Riverside, in 2021. He is currently a Systems Software Engineer at Aruba, a Hewlett Packard Enterprise Company. His research interests include networking, distributed systems, and formal methods.

K. K. Ramakrishnan (Fellow, IEEE) received the M.Tech. degree from the Indian Institute of Science in 1978 and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, USA, in 1981 and 1983, respectively. He is a Professor of computer science and engineering with the University of California at Riverside, Riverside. Previously, he was a Distinguished Member of Technical Staff at AT&T Labs-Research. Prior to 1994, he was the Technical Director and a Consulting Engineer in networking at Digital Equipment Corporation. Between 2000 and 2002, he was at TeraOptic Networks Inc. as the Founder and the Vice President. He has published over 300 papers and has 185 patents issued to his name. He is a Fellow of ACM and AT&T, recognized for his fundamental contributions on communication networks, including his work on congestion control, traffic management, and VPN services.