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

Mohammad Jahanian and K. K. Ramakrishnan
University of California, Riverside, CA, USA. mjaha001@ucr.edu, kk@cs.ucr.edu

Abstract—**In many scenarios, information must be disseminated over intermittently-connected environments when network infrastructure becomes unavailable. Example scenarios include disasters in which first responders need to send updates about their critical tasks. If such updates pertain to a shared data set (*e.g.*, pins on a map), their consistent dissemination is important. We can achieve this through causal ordering and consensus. Popular consensus algorithms, such as Paxos and Raft, are most suited for connected environments with reliable links. While some work has been done on designing consensus algorithms for intermittently-connected environments, such as the One-Third Rule (OTR) algorithm, there is 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' views, 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 basic OTR to support long-term fragmentation and critical decision invalidation scenarios. We integrate the multi-level consistency schema of CoNICE, with a naming schema that follows a topic hierarchy-based dissemination framework, to improve functionality and performance. Performing city-scale simulation experiments, we demonstrate that CoNICE is effective in achieving its consistency goals, and is efficient and scalable in the time for convergence and utilized network resources.**

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, just like 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] better than It 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, the most well-known of which is 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 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 heavily content-oriented network usage of today, has led to the proposal and design of widespread in-network naming frameworks for better-organized information dissemination [15]–[17], showing that with a proper naming schema [18], [19], we can achieve better accuracy and more scalable dissemination in terms of reducing end user and network load, compared

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

to the current address-oriented networking paradigms.

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 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 and 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, 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. The naming component optimizes consensus participation to only users who are relevant and helps with efficiency and scalability. The decision invalidation helps with overcoming the consensus property violations in case of long-term physical fragmentation in the network. CoNICE does not consider mobility as a failure; rather as an asset, helping to deliver messages integral to ensuring consistency across many users. Throughout the paper, we describe and evaluate our architecture and protocol using the use case of first responders geo-tagging with important information on a shared map during a disaster with only an infrastructure-less network.

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 GitHub[1]); 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) Simulation results that show our enhancement leads to a higher degree of agreement among users, with lower overhead.

II. Background and Related Work

Propagation in Intermittently-Connected Environments. There have been a number of works on information propagation in intermittently-connected networks [23]. 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 [24]–[26]. There have also been proposals with additional assumptions that leverage geographical routing and predictions [27]–[29]. Methods such as Bubble Rap [30], dLife [31], SCORP [32], and EpSoc [33] use social data regarding human interactions as the basis of such routing predictions. We use Epidemic Routing [24] 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 [23] (although we can replace it with the some of the other methods mentioned if additional assumptions are reasonable and can be accommodated). In epidemic routing, users buffer messages and upon each encounter, they exchange their Summary Vectors (SV), representing what they have in their buffers. Each node determines what they need from the peer's SV, and requests and initiates message exchanges, avoiding duplicate deliveries [24]. Apart from its benefits, it is observed that epidemic routing has high overhead [23]. 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 [34] 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. GPS can provide accurate time to equipped devices, with negligible skews. However, GPS has several issues that make it a less than perfect method for clock synchronization, especially in decentralized systems. It is vulnerable to spoofing attacks [35], [36], and susceptible to bad weather or lack of line-of-sight (*e.g.*, indoors) [37].They cause innacuracies and excessive usage, especially in "un-assisted" mode. GPS can also greatly reduce battery life [38]. Scalar logical clock [6] defines the relation that message m_1 "happened before" message m_2 ($m_1 \rightarrow m_2$), if they follow FIFO order (some user sends m_1 and then sends m_2), local order (some user receives m_1 and then sends m_2), or a transitivity rule (there exists some message m_3 such that $m_1 \rightarrow m_3$ and $m_3 \rightarrow m_2$) [6]. 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 [39], [40] 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 [41] proposes differential clocks as an optimization to vector clocks, only sending vector differences. [42] proposes that explicitly specifying causality, by sender, helps with scalability. Work in [43] 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.

[1]https://github.com/mjaha/CoNICE

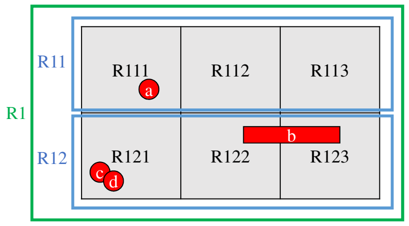

Fig. 1. Example region-ed map with base layer (background) and data layer (pins/shapes)

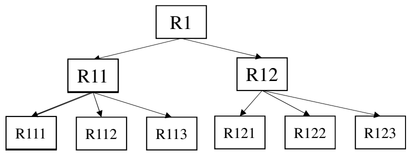

Fig. 2. Namespace pertaining to the map in Fig. 1

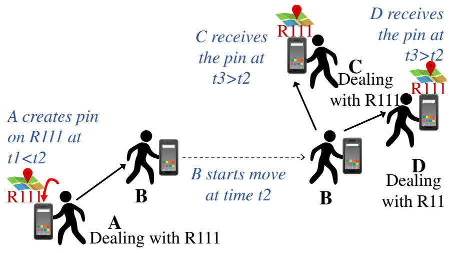

Fig. 3. A scenario overview

Consensus and Strong Consistency. There has been a great deal of work on consensus, the most prominent of which is Paxos [8], [44]. 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. While such solutions work well in connected networks and (partially) synchronous systems (*i.e.*, known upper bound on message latency), it has been shown in [13] that they are not suitable for disconnected environments. Their fault recovery, through Failure Detectors [45], [46], is typically limited to node failures rather than link failures. The Heard-Of model [13] proposes a benign fault model, 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 [47] 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 and adding support for cases where decisions need to be invalidated *e.g.*, due to long-term network fragmentation.

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 [48] of the Information-Centric paradigms, such as in Named Data Networks (NDN) [15]. Work in [49] provides name-based DTN-like dissemination frameworks. However, extra steps are needed for ensuring consistent dissemination. Works in [12], [50] 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 [51] and Secure Scuttlebutt [52] 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 [53] 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. 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. The base layer is the map background, available offline to each user. Informed by the geography pertaining 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. 2. User 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. 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. 3, 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. 3), 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. Our primary goal is to make sure all the users converge to a consistent view of all generated updates on the map data layer, in this disconnected environment, as much as possible.

CoNICE Overview. An overview of the architecture of CoNICE is depicted in Fig. 4. 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

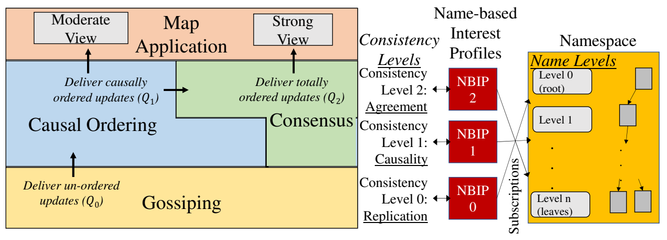
Moderate View
Map Application
Strong View
Deliver causally ordered updates (Q_1)
Deliver totally ordered updates (Q_2)
Causal Ordering
Consensus
Deliver un-ordered updates (Q_0)
Gossiping
Consistency Levels
Consistency Level 2: Agreement
Consistency Level 1: Causality
Consistency Level 0: Replication
Name-based Interest Profiles
NBIP 2
NBIP 1
NBIP 0
Subscriptions
Namespace
Name Levels
Level 0 (root)
Level 1
Level n (leaves)

Fig. 4. Architecture overview of CoNICE

been delivered to individual users. The *Gossiping* component in each user's device is responsible for this function, using Epidemic Routing [24]. Consistency level 1, namely *Causality*, ensures that orderable updates are applied according to their causal relationships and precedence. This 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 [39], [40], 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.

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 [54], similar to [52]; 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 [55] can be leveraged, similar to [56]; 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 [57], which secures content itself, rather than the channel used for delivery.

IV. Naming and Consistency Levels

CoNICE relies on graph-based naming structure and multi-level consistency, both of which we elaborate on, in this section.

A. Graph-Based Naming Framework

The naming schema in CoNICE is designed and represented as a graph structure (*e.g.*, Fig. 2), according to the hierarchical structure of map region-ing: the higher levels in the hierarchy correspond to larger regions. An example of the namespace may be something like "County→City→*etc.*" For mere representation simplicity, we assume each node in the namespace graph has a unique name, so that we do not need to identify a node by mentioning its whole prefix path; *e.g.*, we use "$R111$" instead of "$/R1/R11/R111$".

Region-bound Messages (Publishing). Creation of a message (*e.g.*, an update) that needs to reach interested recipients is a publication in which the region that the message relates to is specified. The region is a name in the namespace and helps with relevancy and selectiveness of dissemination and consistency preservation procedures. A message *belongs to* exactly one region; namely the smallest region large enough to contain that message. For example in Fig. 1, update 'a' belongs to $R111$ while update 'b' belongs to $R12$. To ensure higher coverage during dissemination, we define that an update *covers* region set $\{R_i\}$, containing all *leaf-level* regions that contain that update. For example, update 'a' covers $R111$ (same region it belongs to) while 'b' covers $R122$ and $R123$. The relevance follows the namespace hierarchy. A message that belongs to region R_i is relevant to subscribers of R_i *and* the ancestors of R_i in the namespace. For example, for the namespace in Fig. 2, update 'a' (specified to be in $R111$) should be consistently delivered to all subscribers of $R111$, $R11$ and $R1$.

Name-based Interest Profiles (Subscribing). NBIPs capture the subscriptions of a user locally on their devices. Each NBIP points to one or more nodes (names) in the namespace, and includes the sub-namespaces (as a set of names) below in the hierarchy, rooted at the pointed names. There are three NBIPs: NBIP0 specifies the scope of the user's involvement in the gossiping procedure, NBIP1 for causal ordering procedures, and NBIP2 for consensus sessions. We have $NBIP2 \subseteq NBIP1 \subseteq NBIP0$ for every user.

B. Multi-level Consistency

There are three consistency levels in CoNICE. Users maintain a queue for each region (name) they are interested in for each consistency level, as shown in the example in Fig. 5. All three users A, B, and C in the Fig. are interested in region R_i (we just focus on R_i here, so queues related to other possible regions of the users' interest are not shown). $Q_i^U(R_i)$ denotes the level i queue at user U, and its *slot* number 0, *i.e.*, $Q_i^U(R_i, 0)$, is the first element. The consistency levels are incremental, each feeding the level above it (Fig. 4); elements in Q_0 serve as input for Q_1, and Q_1 serves as input for Q_2.

The level 0 (replication) queue (Q_0) shows the received updates in the order in which they have been received and entered the message buffer (cache) used by the gossiping component. The updates in Q_0 in the Fig, are marked by the update content ('a', 'b', *etc.* shown in bold) and can contain dependencies (*e.g.*, $\frac{b}{a}$ indicates that 'b' depends on 'a'). As seen in the Fig., different users can receive the updates in different orders, as their connectivity is intermittent and there may be no established path between users. In addition to epidemic buffering, this consistency is important as it is the

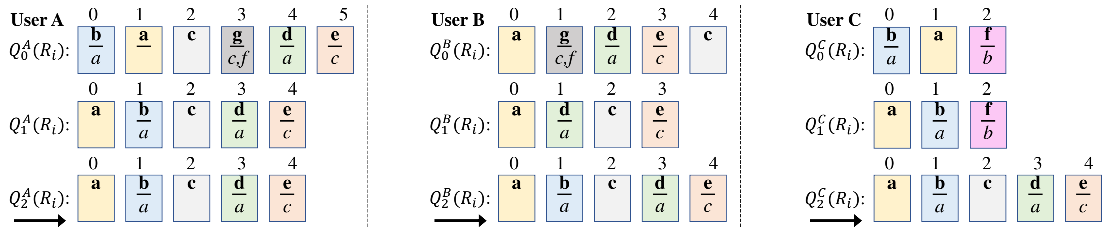

Fig. 5. Example for per- name per- consistency level update queues across three users A, B, and C

starting point for the next level of consistency and for making sure messages are replicated sufficiently to all users in the network, despite the challenging nature of the connectivity.

The level 1 (causality) queue (Q_1) contains the causally ordered updates, respecting the order for the causally orderable updates. As seen in the Fig., updates are only added to Q_1 only after all their dependencies have also been added. For example, for Q_1^A, 'b' appears after 'a', 'e' appears after 'c'. But, 'g' does not appear at all, since one of its dependencies 'f' has not been received at A yet (even though it has reached user C, as seen by event 2 for user C). This is consistently true across all users. However, for un-orderable updates (*i.e.*, those with no causality relation), different users can put them in their Q_1 according to the order with which they were received. For example, updates 'c' and 'd' have no causal relation in the figure and are thus un-orderable; in Q_1^A, 'd' comes after 'c' while in Q_1^B, the reverse is true. Despite the difference, neither ordering is incorrect, as they do not violate causal ordering. Q_1 provides users with a Moderate View, which is useful as it provides a causal, meaningful, and "*a* correct" (rather than "*the* correct") view of the map, even though it might be different from another user's "also correct" Moderate View.

The level 2 (agreement) queue (Q_2) shows the updates in the agreed (by *all* users) order, provided by the consensus procedure. It resolves the differences of different users' Moderate Views regarding un-orderable updates, and provides users with a Strong View in the map that: a) honors the causal ordering, and b) is the same across all different users, as seen in the Fig. Each slot in Q_2 is filled via the result of a distinct consensus *session*. CoNICE first computes Q_1 and then feeds it to the consensus component as initial consensus *contributions* (*i.e.*, vote values). For example, user B's contribution for the second slot of Q_2 is 'd', while users A and C vote for 'b'. Eventually, the majority value ('b' for slot 1) is *decided* and disseminated to everyone. This incremental approach to arriving at consensus provides the following benefits:

1) Consensus starts from a point with potentially more nodes converged on the ordering of a larger number of events, compared to the alternative of jumping from Q_0 to Q_2.
2) Users are eventually provided with a Strong View that respects causal ordering (to perform a meaningful application order of updates), compared to the alternative of completely bypassing and ignoring causal ordering.
3) Users are provided with a useful, causally ordered Moderate View, while waiting for consensus sessions to be completed, compared to the alternative of arriving at a causal ordering after consensus. This is practically important in scenarios such as emergency response, since causal ordering is much less complex and less time-consuming (user-driven) than consensus (community-driven across multiple nodes).

V. Protocols for Consistent Dissemination

In CoNICE, three components, each with their own set of protocols, handle its three consistency levels (Fig. 4), all integrated with naming. The gossiping protocol propagates and replicates messages among users, the causal ordering protocol takes care of causal consistency of delivered updates, and consensus protocol ensures agreement and strong consistency.

A. Gossiping Protocol

Gossiping makes the propagation of messages possible in intermittent connectivity, thus helping with level 0 consistency (replication) in CoNICE. Our layering provides a separate abstract interface, making the choice of the gossiping mechanisms independent of higher-level consistency mechanisms and vice versa. We use epidemic propagation [24] as justified in §II; 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 message matching their interest, 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 in 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 indicated to subscribe to the R_B in the namespace hierarchy, to be able to also participate in its level 1 and level 2 procedures. To further reduce the cost of epidemic propagation, CoNICE uses hop count limits and cleaning buffers of obsolete messages (similar to [12], [24]).

B. Causal Ordering Protocol

To ensure causal consistency, CoNICE uses and extends the Vector Clock (VC) [39] method, mainly that we limit the notion of causality to those updates that belong to the same region. This way, thanks to CoNICE's naming schema, the number of causal prerequisites to fulfill decreases significantly from the whole set of update space, to a selective set of "only relevant" updates, thus helping with better scalability of causal ordering.

Different updates that belong to the same region can potentially depend on each other. As an example, in Fig. 1, 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's gossiping module.

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      $UID_A \leftarrow$ id of this user A
5      $M_A \leftarrow$ set of updates created by A
6      $M_{Q1} \leftarrow$ set of all updates at level 1 queues at A
7      $UpID_R \leftarrow \{\}$ /* set of reference updates to be included */
8  $seqNum \leftarrow nextSeqNum(R_B, M_A)$
9  foreach $user\ B\ in\ creators(M_{Q1})$ do        /* identify references */
10     $UpID_R \leftarrow UpID_R \cup latestUpdate(B)$
11 foreach $u \in UpID_R$ do        /* make reference list more compact */
12     if $\exists u' \in UpID_R : u \rightarrow u'$ then $UpID_R \leftarrow UpID_R - \{u\}$
13 $msg \leftarrow \langle UPDATE, UID_A, R_B, seqNum, R_C, UpID_R, data\rangle$
14 $publish\, msg$ /* send to gossiping module */
15 $M_A \leftarrow M_A \cup msg$
16 $M_{Q1} \leftarrow M_{Q1} \cup msg$
```

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      $UID_A \leftarrow$ id of this user A
7      $M_{Q0} \leftarrow$ set of all updates at level 0 queue at A
8      $M_{Q1} \leftarrow$ set of all updates at level 1 queues at A
9      $missing \leftarrow \{\}$ /* update IDs of missing prerequisites */
10 Upon receive
     ($msg$=$\langle UPDATE, UID_C, R_B, seqNum, R_C, UpID_R, data\rangle$) 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 $R_B \in NBIP1_A$ then
14         foreach $u' \in M_{Q0} \wedge u' \notin M_{Q1}$ do
15             if $dependencies(u')\ satisfied$ then $M_{Q1} \leftarrow M_{Q1} \cup \{u'\}$
16         $missing \leftarrow missing \cup missingImplicit(u, M_{Q1})$
17         $missing \leftarrow missing \cup missingExplicit(u, M_{Q1})$
18         if $missing = \{\}$ then $M_{Q1} \leftarrow M_{Q1} \cup \{u\}$
19         foreach $UpID_i \in missing$ do $publish\langle REQUEST, UpID_i\rangle$
20 Upon receive ($msg$=$\langle RESPONSE, UID_R, UID_C, R_B, seqNum,$
     $R_C, UpID_R, data\rangle$) do
21     procUpdate($UID_C, R_B, seqNum, R_C, UpID_R, data$)
22     if $UID_R = UID_A$ then $cancel\, msg$
```

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"). 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 (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 disconnected environment with gossiping, this may lead to starvation and indefinite waiting, since "gossips may die out" [25]. 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 [58]):

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.

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,

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      $Q_1^A \leftarrow$ user A's current level 1 queue
6      $Q_2^A \leftarrow$ user A's current level 2 queue
7      $UID_A \leftarrow$ id of this user A
8      $contribs_s \leftarrow \{\}$        /* contributions multiset at A */
9      $decs_S = \langle DECISION, UID_D, R_S, s_S, a_D, n_D, v_D \rangle \leftarrow \{\}$
10     $solved_S \leftarrow false$ /* as $dec_S$ is empty initially */
11     $v_I \leftarrow Q_1^A(R_S, s_S)$ /* Noop if null */
12     if $v_I \neq Noop$ then startAttempt$(1, 1, v_I)$
13 Procedure startAttempt$(a, r, v)$
14     $v_S \leftarrow v$
15     $a_S \leftarrow a$
16     startRound$(a_S, r)$
17 Procedure startRound$(r)$
18     $r_S \leftarrow r$
19     $publish\, msg$=$\langle CONTRIBUTION, UID_A, R_S, s_S, a_S, r_S, n_s, v_s \rangle$
20     $contribs_S \leftarrow contribs_S \cup msg$
21 Upon receive $(msg$=$\langle CONTRIBUTION, UID, R_S, s_S, a, r, n, v \rangle)$ do
22     if $R_S \notin NBIP2_A$ then $cancel\, msg$
23     switch $a$ do
24         case $a > a_S$ do
25             foreach $m \in contribs_S$ do $cancel\, and\, delete\, m$
26             $n_S \leftarrow max(n_S, n)$
27             $contribs_S \leftarrow \{msg\}$
28             startAttempt$(a, r)$
29         case $a < a_S$ do $publish\, D_S$
30     if $solved_S$ then
31         $publish\, D_S$
32         $cancel\, and\, delete\, msg$
33     switch $r$ do
34         case $r > r_S$ do
35             foreach $m \in contribs_S$ do $cancel\, and\, delete\, m$
36             $n_s \leftarrow max(n_S, n)$
37             $contribs_S \leftarrow \{msg\}$
38             startRound$(r)$
39         case $r < r_S$ do $cancel\, and\, delete\, msg$
40         case $r = r_S$ do
41             $contribs_S \leftarrow contribs_S \cup msg$
42             $n_s \leftarrow max(n_S, n)$
43             if $|contribs_S| > (2/3) \times n_S$ then
44                 $v_S \leftarrow smallest\, most\, frequent\, non\text{-}Noop\, in\, contribs_S$
45                 if $all\, equal\, to\, \overline{V}\, in\, contribs_S\;\; excluding\, Noop$
                   then decide$(R_S, s_S, a_S, v_S)$
```

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 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

Algorithm 4: Consensus: Decisions

```
1  Procedure decide$(UID, R_S, s_S, a, n, v)$
2      if $\neg solved_S$ then
3          if $\exists s' \neq s_S \wedge Q_2^A(R_S, s') = v$ 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          $solved_S \leftarrow true$
7          $v_S \leftarrow v$
8          foreach $m \in contribs_S$ do $cancel\, and\, delete\, m$
9          $contribs_S \leftarrow \{\}$
10         $publish\, msg = \langle DECISION, UID_A, R_S, s_S, a, n, v_S \rangle$
11         $D_S \leftarrow msg$
12     else                                   /* need to invalidate */
13         if $msg \neq D_S$ then
14             if $a = a_D$ then
15                 if $n > n_D$ then $v'_D \leftarrow v$
16                 else if $n < n_D$ then $v'_D \leftarrow v_D$
17                 else if $n = n_D$ then
18                     if $v \geq v_D$ then $v'_D \leftarrow v$
19                     else if $v < v_D$ then $v'_D \leftarrow v_D$
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         $Q_2^A(R_S, s_S) \leftarrow v$
25         foreach $v' \in Q_2^A(R_S)$ $that\, violates\, causality\, with\, v$ do
26             $reorder\, locally\, through\, deterministic\, sort$
27 Upon receive $(msg = \langle DECISION, UID, R_S, s_S, a, n, v \rangle)$do
28     if $R_S \notin NBIP2_A$ then $cancel$
29     decide$(UID, R_S, s_S, a, n, v)$
```

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 [59]), 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. 2. 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

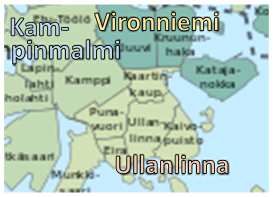

Fig. 6. Map of our Helsinki-based simulation scenario

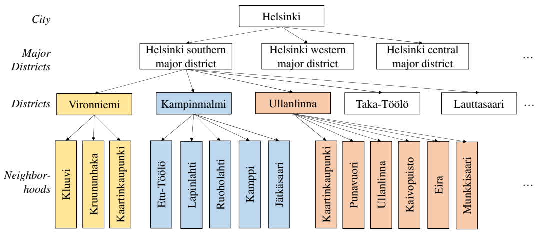

Fig. 7. Namespace for our Helsinki-based simulation scenario

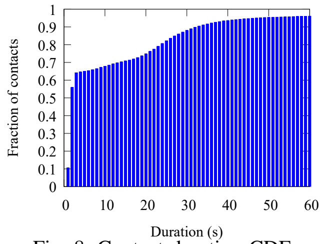

Fig. 8. Contact duration CDF

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's consensus protocol satisfies the following properties:

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

Property 3. *Total and Causal Order of the Strong View*. If user A applies update u to its moderate view at $<R, slot>$, then eventually, for any other correct user B that applies update u' to its moderate view at $<R, slot>$, we have u=u'. Additionally, the Strong View at any user A, respects causal order.

In normal situation, supported by basic OTR, the above properties are easy to prove [13]. Due to the nature of the environment and extended assumptions we consider, there are additional cases where the properties may get violated, which we provide remedies for in Alg. 4 (proofs in [58]):

1. *Loss of causality*. Due to the lack of central orchestration and coordination in CoNICE, there may be cases that decided values for Q_2 may not respect causality; *e.g.*, having v_i for $<R, s_i>$ and v'_i for $<R, s_{i+k}>$ while $v'_i \rightarrow v_i$ (*i.e.*, v_i depends on v'_i). We can completely prevent this by running consensus sessions one by one, sequentially. However, this is not efficient and can lead to starvation, especially considering how time-consuming a consensus session can be. Thus, we provide a pragmatic solution to recover from this violation. In case this happens, the user swaps the values between slots, through a deterministic sorting algorithm (lines 25–26). This invalidation can be repaired entirely locally, without further messages.

2. *Long-term physical fragmentation*. Sometimes, more "intense" cases of fragmentation can occur, going beyond the "good periods" assumption of OTR: assume two disconnected shelters in a region, each with a number of users trapped in them, with no mules or paths between the two shelters for a very long time (much longer than consensus durations). As a result of such "long-term" fragmentation, users independently beacon, create, disseminate, and solve consensus within their partitions. In case a path appears subsequently between the two shelters (*i.e.*, by a mule), and messages get exchanged between the two, the network will include two decisions with different values for the same $<R, s>$ pair. This will violate the correctness of consensus. To remedy this, we make use of *UID* and n fields of decision messages to invalidate decisions already-made (lines 12–23), and upgrade the decisions of users to one from the fragment with the higher population (and in case of tie, the one with a higher user ID). This invalidation can be repaired within the same attempt.

3. *Duplicate decisions*. There may be cases where the same value is picked for two different slots; *e.g.*, having v_i for $<R, s_i>$ and v_i for $<R, s_{i+k}>$. This shows that for some reason, one value was picked for two slots. This can be caused by divergences in users' initial values and is not fixed by the basic OTR. Thus, there will be a value that was missed during the consensus rounds, and got replaced by the same value picked in another session. As a result, consensus has to restart, albeit as the next attempt. The user detecting the two duplicates, will keep the first one (in this example, s_i), and starts a new session to re-do consensus for the second slot (s_{i+k}). The only catch is, if the initial value of the user for s_{i+k} (*i.e.*, from its Q_1) is v_i, it will pick 'Noop'. Otherwise, that user will pick $Q_1(R, s_{i+k})$. This way, the consensus will be performed again, giving v_i less chance to be picked for s_{i+k} at the end (lines 3–5). This invalidation can be repaired with a new attempt.

VI. Evaluation

To evaluate CoNICE, we perform a simulation based on a partial map of the city of Helsinki (Fig. 6, [60]) using the ONE simulator [61]. The associated hierarchically-structured namespace (Fig. 7) follows the "City→Major districts→Districts→Neighborhoods" structure. Our simulation environment consists of the three districts (and hierarchically, the neighborhoods in them) highlighted in the Fig., and is 4500×3400 meters large. 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

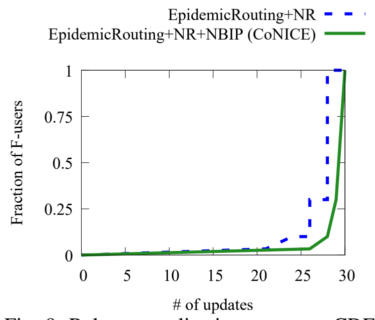

Fig. 9. Relevant replication coverage CDF

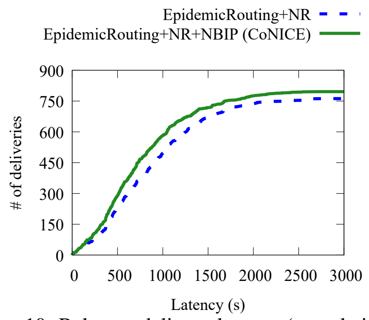

Fig. 10. Relevant delivery latency (cumulative)

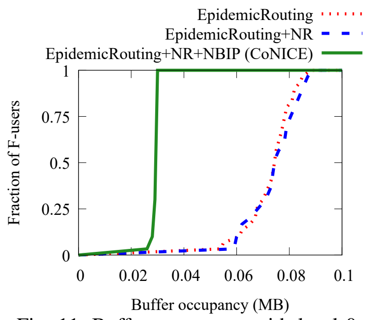

Fig. 11. Buffer occupancy with level 0

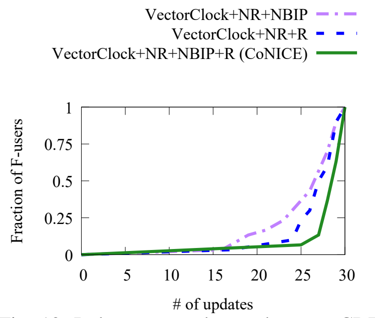

Fig. 12. Relevant causal completeness CDF

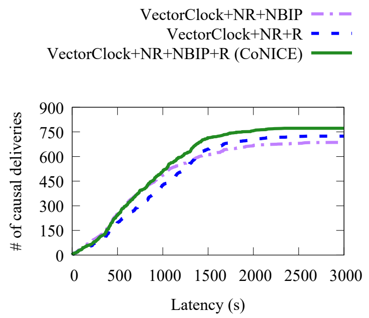

Fig. 13. Relevant causal delivery (cumulative)

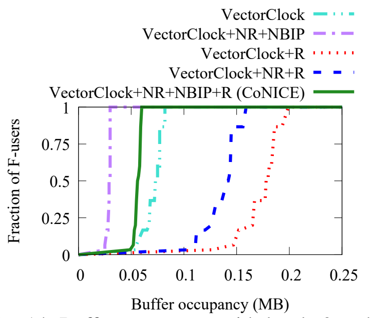

Fig. 14. Buffer occupancy with levels 0 and 1

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

not participate in any consensus sessions. 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. 8. As the Fig. 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. Fig. 9 shows the CDF of RC_{rel}. As Table I and Fig. 9 show, CoNICE achieves

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

better RC_{rel} than 'epidemic routing+NR' (NR is name-based region-ing for publications), as it adds name-based interest profiling (NBIP). Fig. 9 shows that higher percentage of F-users have received higher number of updates with CoNICE. It also achieves better latency with more relevant deliveries (RL_{rel}), as shown cumulatively in Fig. 10 (at most can reach 30×30=900). This is due to naming which makes relays and queued messages more useful and relevant. This is also shown in Fig. 11, which shows the buffer occupancy (MB) across all F-users at the end of the simulation. Also, basic epidemic routing that uses no naming (thus, the notion of 'relevancy' is 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.

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. 12 shows the CDF of relevant causal completeness (CC_{rel}), which denotes how many of updates have been causally applied at F-users. As the Fig. 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. 13

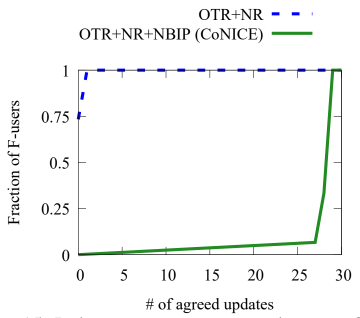

Fig. 15. Relevant agreement completeness CDF

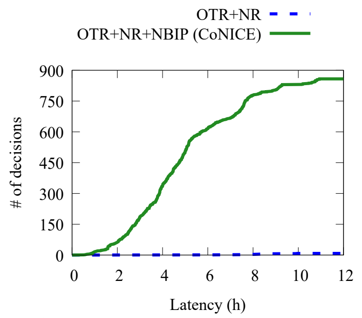

Fig. 16. Relevant decision latency (cumulative)

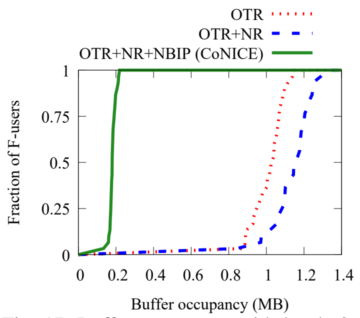

Fig. 17. Buffer occupancy with levels 0–2

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

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 appendices more selective, having to only depend on relevant updates. Fig. 14 shows CoNICE achieves better buffer usage than most of the 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. 15 (CDF of AC_{rel}) show, CoNICE achieves a dramatically better agreement completeness. Fig. 15 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 (as example in Fig. 5). Fig. 16 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. 17. 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.

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. Our simulation experiments on an application of map-based geo-tagging in emergency response shows 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.

VIII. Acknowledgements

This work was supported by the US Department of Commerce, NIST (award 70NANB17H188) and US NSF grant CNS-1818971. We thank our shepherd, Prof. Rute Sofia, for her support and the reviewers for their valuable comments.

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 *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2018, pp. 79–84.

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 27–34.

[3] C. Boldrini, K. Lee, M. Önen, J. Ott, and E. Pagani, "Opportunistic networks," *Computer Communications*, no. 48, pp. 1–4, 2014.

[4] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in lte-advanced networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 1923–1940, 2014.

[5] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming*, 2nd ed. Springer Publishing Company, Incorporated, 2011.

[6] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, p. 558–565, Jul. 1978.

[7] D. R. Cheriton and D. Skeen, "Understanding the limitations of causally and totally ordered communication," in *Proceedings of the fourteenth ACM symposium on Operating systems principles*, 1993, pp. 44–57.

[8] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.

[9] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIX ATC'14)*, 2014, pp. 305–319.

[10] M. Swan, *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[11] F. Borran, R. Prakash, and A. Schiper, "Extending paxos/lastvoting with an adequate communication layer for wireless ad hoc networks," in *2008 Symposium on Reliable Distributed Systems*. IEEE, 2008, pp. 227–236.

[12] A. Benchi, P. Launay, and F. Guidec, "Solving consensus in opportunistic networks," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, 2015, pp. 1–10.

[13] B. Charron-Bost and A. Schiper, "The heard-of model: computing in distributed systems with benign faults," *Distributed Computing*, vol. 22, no. 1, pp. 49–71, 2009.

[14] 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, pp. 1–12.

[15] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, 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 *2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*. IEEE, 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 *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 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 *2016 IEEE international symposium on local and metropolitan area networks (LANMAN)*. IEEE, 2016, pp. 1–6.

[19] C. Ghasemi, H. Yousefi, K. G. Shin, and B. Zhang, "On the granularity of trie-based data structures for name lookups and updates," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 777–789, 2019.

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

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

[22] F. Houshmand and M. Lesani, "Hamsaz: replication coordination analysis and synthesis," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–32, 2019.

[23] K. K. Ahmed, M. H. Omar, and S. Hassan, "Survey and comparison of operating concept for routing protocols in dtn," *Journal of Computer Science*, vol. 12, no. 3, pp. 141–152, 2016.

[24] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," 2000.

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

[26] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005, pp. 252–259.

[27] U. G. Acer, S. Kalyanaraman, and A. A. Abouzeid, "Weak state routing for large-scale dynamic networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1450–1463, 2010.

[28] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *International Workshop on Service Assurance with Partial and Intermittent Resources*. Springer, 2004, pp. 239–254.

[29] J. Borah, "Application of computational intelligence paradigm in the probabilistic routing for intermittently connected network," *IOSR J. Comput. Eng*, vol. 8, pp. 32–36, 2012.

[30] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576–1589, 2010.

[31] W. Moreira, P. Mendes, and S. Sargento, "Opportunistic routing based on daily routines," in *2012 IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*. IEEE, 2012, pp. 1–6.

[32] ——, "Social-aware opportunistic routing protocol based on user's interactions and interests," in *International conference on ad hoc networks*. Springer, 2013, pp. 100–115.

[33] H. Lenando and M. Alrfaay, "Epsoc: social-based epidemic-based routing protocol in opportunistic mobile social network," *Mobile Information Systems*, 2018.

[34] R. Koch, R. Moser, and P. Melliar-Smith, "Global causal ordering with minimal latency," in *International Conference on Parallel and Distributed Computing and Networking*, 1998, pp. 262–267.

[35] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 75–86.

[36] P. Papadimitratos and A. Jovanovic, "Gnss-based positioning: Attacks and countermeasures," in *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE, 2008, pp. 1–7.

[37] H. K. Alper, "Network time with a consensus on clock," Cryptology ePrint Archive, Report 2019/1348, 2019, https://eprint.iacr.org/2019/1348.

[38] A. Dimri, H. Singh, N. Aggarwal, B. Raman, K. K. Ramakrishnan, and D. Bansal, "Barosense: Using barometer for road traffic congestion detection and path estimation with crowdsourcing," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 1, pp. 1–24, 2019.

[39] J. FIDGE, "Timestamps in message-passing systems that preserve the partial ordering," in *Proc. 11th Australian Comput. Science Conf.*, 1988, pp. 56–66.

[40] F. Mattern, "Virtual time and global states of distributed systems," *Parallel and Distributed Algorithms*, pp. 215–226, 1989.

[41] M. Singhal and A. Kshemkalyani, "An efficient implementation of vector clocks," *Information Processing Letters*, vol. 43, no. 1, pp. 47–52, 1992.

[42] P. Bailis, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "The potential dangers of causal consistency and an explicit solution," in *Proceedings of the Third ACM Symposium on Cloud Computing*, 2012, pp. 1–7.

[43] R. J. de Araújo Macêdo, "Causal order protocols for group communication," in *Proc. of Brazilian Symp. nn Computer Networks (SBRC)*, 1995.

[44] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, 2001.

[45] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.

[46] E. Gafni, "Round-by-round fault detectors (extended abstract) unifying synchrony and asynchrony," in *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, 1998, pp. 143–152.

[47] F. Brasileiro, F. Greve, A. Mostéfaoui, and M. Raynal, "Consensus in one communication step," in *International Conference on Parallel Computing Technologies*. Springer, 2001, pp. 42–50.

[48] M. Jahanian and K. K. Ramakrishnan, "Name space analysis: verification of named data network data planes," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 44–54.

[49] 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 *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 9–18.

[50] A. Datta, S. Quarteroni, and K. Aberer, "Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks," in *International Conference on Semantics for the Networked World*. Springer, 2004, pp. 126–143.

[51] T. Li, Z. Kong, S. Mastorakis, and L. Zhang, "Distributed dataset synchronization in disruptive networks," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2019, pp. 428–437.

[52] D. Tarr, E. Lavoie, A. Meyer, and C. Tschudin, "Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 1–11.

[53] L. Wang, Y. Lyu, J. Liu, W. Shang, W. He, D. Wang, and G. Min, "Naxos: A named data networking consensus protocol," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2018, pp. 986–991.

[54] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[55] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *SP*, 2007.

[56] C. A. Lee, Z. Zhang, Y. Tu, A. Afanasyev, and L. Zhang, "Supporting virtual organizations using attribute-based encryption in named data networking," in *CIC*, 2018.

[57] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE communications surveys & tutorials*, vol. 20, no. 1, pp. 566–600, 2017.

[58] M. Jahanian and K. K. Ramakrishnan, "Conice: Consensus in intermittently-connected environments by exploiting naming with application to emergency response," University of California, Riverside, Tech. Rep., 2020. [Online]. Available: https://www.cs.ucr.edu/∼mjaha001/CoNICE-TR.pdf

[59] D. Cavin, Y. Sasson, and A. Schiper, "Consensus with unknown participants or fundamental self-organization," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2004, pp. 135–148.

[60] Wikipedia, "Subdivisions of helsinki." [Online]. Available: https://en.wikipedia.org/wiki/Subdivisions_of_Helsinki

[61] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*, 2009, pp. 1–10.