Collaborative Editing over Opportunistic Networks: State of the Art and Challenges

Noha Alsulami*§ and Asma Cherif§
§ Faculty of Computing and Information Technology, IT department
§King Abdulaziz University

*Faculty of Computing and Information Technology, IT department
*University of Jeddah, AlKamel Branch
*§ Jeddah, Saudi Arabia

***Abstract*—Emerging Opportunistic Networks (ON) are under intensive research and development by many academics. However, research efforts on ON only addressed routing protocols as well as data dissemination. Too little attention was given to the applications that can be deployed over ON. These are assumed to use immutable data (*e.g.*, photos/video files). Nevertheless, Collaborative Editors (CE) which are based on mutable messages are widely used in many fields. Indeed, they allow many users to concurrently edit the same shared document (*e.g.*, Google Docs). Consequently, it becomes necessary to adapt CE to ON which represents a challenging task. As a matter of fact, CE synchronization algorithms should ensure the convergence of the shared content being modified concurrently by users. In this work, we give an overview on ON and CE in an attempt to combine both states of the art. We highlight the challenges that could be faced when trying to deploy CE over ON.**

***Keywords*—Opportunistic networks; collaborative editors; consistency maintenance; operational transformation approach**

I. Introduction

With the increase of mobile devices use in the last years, Opportunistic Networks (ON) have become an important research field. Indeed, ON allow to ensure wireless communication between peer nodes in a flexible and high dynamic way. As a matter of fact, any node can join and leave the network at any time. The communication paths between senders and receivers are neither direct nor static since the network's topology can change frequently and dynamically according to nodes' movements [1], [2]. ON rely on the Store-Carry-and-Forward (SCF) approach [3] to transmit data between nodes. Each node has a range of neighbors to which it can forward messages. To reach a destination that resides outside the sender's range, the latter simply forwards the message to direct contacts. These take then the responsibility to forward the message using an opportunistic communication pattern till it reaches the destination [1], [2], [4]. Consequently, the communication between nodes is made simpler.

Most importantly, ON demonstrate to be effective in emergency situations caused either by natural catastrophes or even terrorist attacks where the network infrastructure is made unavailable or broken. It may be also the only way of ensuring communication in poor countries like India. In such situations, it is very important to allow users to be interconnected for security and rescue reasons [1]. This could be achieved through the establishment of an ad hoc communication using mobile devices that are massively available nowadays.

Collaborative Editors (CE) provide a set of shared documents that may be modified at any time by geographically dispersed users [5], [6]. This kind of applications may be used in different situations where ON are established between many users. For instance, it is useful for participants during academic events (*e.g.*, conferences) or rescue teams in emergency situations to face the breakdown of the communications infrastructure.

To illustrate the importance of CE in ON, we consider the example depicted in Fig. 1 where a university faces a fire incident. We suppose that the infected area is separated into two operation sections: emergency area (EA) and First-Aids Area (FAA). EA represents the university buildings affected by the incident while FAA is reserved to give first aids to patients who will be then transported by ambulances towards hospitals according to their priorities and current status. It is crucial to record and share patients and victims' information during and after the incident between the operation areas and also between rescue team members and hospitals. Thus, all required information such as the patients' status and what kind of first aids and medications they have received are available before their arrival at hospitals. Therefore, it is obvious that using CE instead of a pen-and-paper approach will improve the efficiency of the rescue operation. Indeed, it is easy to equip all rescue members with inter-connected mobile devices. Meanwhile, the communication infrastructure for such situations can't rely on traditional networks since they will be probably out of use during the incident. Thus, ON represent a very appropriate communication means since it only requires the use of Wi-Fi-equipped mobile devices.

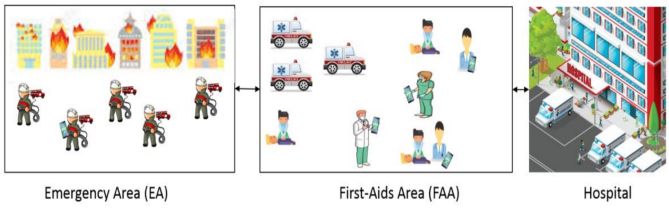

Fig. 1. Motivating the use of CE over ON.

However, the data handled by ON is generally supposed to be non modifiable like photos or video files. Too little attention was given to modifiable content over ON. Indeed, few researches have focused on how to adapt applications dealing with mutable content to ON [1], [7]. Indeed, combining CE with the high dynamic and mobility aspects of ON represents a real challenge. Moreover, ON require low storage capacity and low energy consumption.

Our paper aims at combining the state of the art of both CE and ON approaches and presenting the challenges faced when applying CE over ON.

First, this adaptation is motivated by the noticeable shift from desktop to mobile devices (laptops and smartphones, etc.). This phenomenon has lead to high mobility which is the basic motivation of ON. Since ON represent a new kind of networks, there is an urgent need to adapt many types of applications to ON.

Secondly, CE are more and more used nowadays due to their support of collaborative work, thus being useful for wide range of users in many fields. Adapting CE to ON is worthwhile because they allow opportunistic and ad-hoc teams to collaboratively edit the shared documents in a flexible way. Most importantly, CE are relevant for rescue teams. In [8], the need of adequate software tools in triage management was motivated which can be easily achieved by CE. Thus, CE are useful for disaster scenarios such as natural catastrophes ($e.g.$, hurricanes, earthquakes, etc.) or even human-generated network breakdowns. In fact, it such situations it is difficult to distribute critical notifications and rescue information to and among citizens [9]. CE can serve as a means of communication and collaboration in such situations.

Thirdly, CE should be allowed in ON since they are the only possible communication means in emerging countries that are disconnected from the global network ($e.g.$, countries in Africa or India) [1].

Finally, an interest to adapting CE over ON [1], [7] has emerged during the last years. However, the proposed solutions have many limitations. Thus, it is important to highlight the challenges faced when deploying CE over ON to help CE designers in taking the appropriate deployment and testing choices.

The main contribution of this survey is to provide an accurate study of both ON and CE. Therefore, we propose a set of comparison criteria between CE that take into account a possible deployment over ON. We highlight the challenges that might arise when adapting CE to an opportunistic context. To our knowledge, this is the first attempt to combine the literatures of CE and ON to provide a basic reference for CE designers and developers.

The remainder of this paper is organized as follows: First, we give an overview over ON in Section II. Second, we discuss CE types and properties in Section III. Afterward, we survey the most significant OT-based collaborative editing algorithms and compare between them in Section IV to raise the challenges that can be faced over ON. In Section V, we give an overview on existing CE models adapted to ON. Finally, we conclude in Section VI.

II. Overview on Opportunistic Networks

ON are a kind of Delay Tolerant Networking (DTN) which has become popular in environments such as developing countries or disasters areas. Indeed, DTN is designed to support the disruption of connectivity and long delivery delays [10]. Precisely, ON are wireless ad hoc networks that represent an extension of mobile ad hoc networks [2]. ON represent an emergent solution for establishing communication even in the absence of any network infrastructure where routing depends on contact opportunities [10].

The key feature of ON is to be independent of any fixed network infrastructure. They ensure intermittent communication between peer nodes (or users) in a highly dynamic fashion, $i.e.$, there is no fixed topology and any node can join and leave the network at any time. More precisely, the sender and receiver do not need to be directly interconnected [1], [2].

A. Features of Opportunistic Networks

In ON, nodes are mobile devices with wireless networking capability [11]. There are three kinds of nodes: source, destination and intermediate. Indeed, ON nodes carry messages, store them in their local memories then forward them when a relevant opportunity occurs [10]. In ON, contacts are unpredictable since nodes do not have any knowledge about which node will enter in their ranges and when [10]. The message can be forwarded in ON when the nodes get an opportunity to send it ($i.e.$, when the intermediate nodes come in node's range that enables the sender to forward the message) [2]. Each node has a range of nodes with which it can communicate. When a sender node wants to send a message to any destination that resides outside its range, it relies on one of the closest intermediate nodes, $i.e.$, direct contacts. Next, the message is carried by and stored in intermediate nodes that wait for the appropriate opportunity to forward the message hop by hop till it reaches its destination. This process repeats until the delivery of the message [2].

ON represent a very appropriate communication means for scenarios where the network infrastructure is unavailable such as in sparsely populated areas. It is also very useful in large scale disasters due to the damage of network infrastructures. Finally, it may be the only communication mean in urban scenarios where the cellular network is overloaded or worse non-existing [12].

B. Deployment Constraints and Challenges

ON networks are characterized by the absence of any infrastructure. There are no predictable and fixed communication paths between nodes. Instead of direct transmission, the Store-Carry-and-Forward (SCF) approach [3] is used to transmit data and relies on the nodes mobility. Thus, the deployment constraints in ON are [4]:

- The links between nodes are temporary, so the network topology is highly dynamic [13].
- The routes are established dynamically by intermediate nodes that play the role of routers.
- The network is heterogeneous since it may house different kinds of devices ($e.g.$, cell phones, laptops, sensors, cameras, etc.).

- The connectivity is intermittent due to the high mobility of ON, the node can move, join and leave the network at any time which leads to network partitioning [14].

Accordingly, there are many challenges that need to be considered when designing an application to be deployed over ON including [2]:

- **High Mobility**: This leads to a lack of any previous knowledge about network information since any node can move in, join and leave the network freely.
- **Unpredictable Contact**: Any node in ON may contact any other node unpredictably due to the high mobility of nodes.
- **Storage constraint**: Intermediate nodes between source and destination require to have enough storage space for storing both routing and application messages until they make an opportunistic contact with another node or with the final destination.
- **Energy**: Managing energy in ON is a technical challenge due to the cost introduced by data transmission between nodes. Energy consumption rate increases when there are multiple replicated messages [15].

C. Routing Protocols

In ON, routes are built dynamically from the source node to the destination. Any node can be used opportunistically to deliver messages to the appropriate destination [10]. There are two approaches for routing in ON namely forwarding-based approaches and flooding-based approaches [2]:

The forwarding-based approach depends on the knowledge acquired by a node to take the decision of whether a message should be forwarded or not and to which node. This approach includes the following routing models [2]:

- **Direct transmission**: It consists in carrying the message by the sender itself till it meets the destination. It is a simple and trivial approach and reduces the communication overhead at the expense of very long delays.
- **Location-based**: It consists in exploiting additional information to forward a message. These information may include the proximity to the destination ($e.g.$, MobySpace protocol [2]).
- **Knowledge-based**: It consists in selecting the forwarding nodes based on its context information ($e.g.$, Context Aware Routing (CAR)).

As for the flooding-based approach, it is based on broadcasting messages to all neighbor nodes and it has two types:

- **Epidemic routing** [16]: It uses pair-wise exchange of messages between the nodes [2]. It is a well-known routing protocol that replicates all messages carried by any node to all the other nodes coming into its contact. It achieves high delivery probability since many nodes carry a copy of each message at the expense of network congestion. Epidemic with ACK is one of Epidemic variations and it uses acknowledgment messages generated when the message is delivered to its destination. It eliminates multiple copies of any message once its ACK is received by the source. However, this version produces additional traffic overhead [10]. To avoid congestion due to flooding, the next family of protocols were proposed.
- **Estimate/prediction routing**: In this approach, estimation and prediction information like contact probabilities are used to decide about message forwarding [2]. The most known routing protocol that uses estimate/prediction routing is the Probabilistic Routing Protocol using History of Encounter and Transitivity (PRoPHET) [17]. It forwards messages based on encounters to estimate the probability of a given node to deliver a messages. Each node stores locally a probability value that is exchanged and updated with each contact. Thus, the message is only forwarded to nodes with higher contact probability with destination [10].

D. ON Applications

In the literature of ON, the applications that are well suited for an opportunistic context are [2]:

- **Recommender systems**: They give recommendations on various items by using the information collected from tracking user activities and mobility patterns.
- **Opportunistic computing**: To perform distributed tasks in ON, this application uses shared services, resources and applications.
- **Crisis Management (Emergency Applications)**: ON are very appropriate to be used in emergency situations when there are unexpected disruptive events leading to the breakdown of traditional networks. Ensuring the messages and data generated in the disaster area is the most important objective in emergency cases in order to reach their destination. This is done without any loss and these messages contain information about victims as well as information for the global coordination of the emergency response [10].
- **Pervasive healthcare**: ON can be used to create an intelligent system in order to track patients as well as to monitor different parameters either physical or physiological.

E. Testing ON applications

To conduct tests and evaluate any application and/or routing protocol over ON, a testing framework is required. The most used simulation tools in ON are ONE [18] and MobEmu [19]. Both were used to test CE over ON in [1], [7]. Testing CE over ON by means of simulations requires to use either an appropriate mobility model or a real mobility trace.

Mobility Models. Node movements can increase the probability of message delivery and the opportunity for communication between the source and the destination. These movements are implemented through mobility models which are set of algorithms and rules that define the node movement patterns.

Mobility models are generally used to provide performance results and design high performance routing protocols. Indeed, they simulate the behavior of real mobile nodes and are used instead of real mobility traces that are more difficult to handle [20]. Examples of mobility models are:

- **Random Waypoint (RWP)**: In this model, there is no restrictions on node mobilities where a random path is selected to forward the message to the destination. It creates zigzag paths within the network area because each node moves direct with a constant speed from the starting location to the next location. There is no algorithm used to select the shortest path where this results in taking less time to forward the message [15], [20].
- **Map based movement model**: In this model, nodes can move based on predefined maps for real cities where packets are forwarded to the destination depending on the path defined by the map [15].
- **Shortest Path Map Based Movement (SPMBM)**: In this model, the node can specify the next destination by selecting any random point on the map. Then, the shortest path algorithm Dijkstra is used in order to find the shortest path to that selected point. This model is the best mobility model since nodes select the shortest path towards the destination [15], [20].

In CE area, the mobility models Modified Random Direction [21] and SMOOTH [22] have been used to test a revision control management system as a kind of CE [7].

Mobility Traces. Mobility traces reflect the real nature of vehicular and human mobility. They are used to validate new applications and protocols [23]. They may also include the node contacts and information about nodes' interests [1].

Though mobility traces are very similar to real movement patterns, they introduce a high deployment cost and time overhead in contrast to mobility models [23]. As for mobility models they allow to test a very high number of nodes thus achieving scalability compared to mobility traces [23].

Many real mobility traces were captured to test ON [23], but the most used trace in ON literature is Infocom [24]. In the area of collaborative editing, Infocom was used to test the work of [1] in addition to Sigcomm [25] and UPB [26] mobility traces.

After we have explained the principle of ON, we present in the following an overview on CE as useful and famous kind of applications that needs to be adapted and deployed over ON. We also discuss CE properties and problems.

III. Overview on Collaborative Editors

CE have many benefits including improving the final result by reducing errors, getting different viewpoints and skills as well as shorting the production time of the final document [27]–[29]. CE are mainly used by communities that produce reports including scientists collaborating on a research project [27], software engineering teams designing and implementing software systems, contributors to wiki pages edition or musicians producing music scores, etc. As famous CE examples, we cite Google Docs that enables many users in different locations to simultaneously collaborate on the same document [30] and Wikipedia that allows to write collaboratively the largest shared knowledge database.

According to the communication type offered by a Computer Supported Collaborative Work (CSCW) system, there are two kinds of editing systems [30], [31] (see Fig. 2):

- **Synchronous**: They allow users cooperation in real-time fashion, carry out the shared objects' updates and broadcast them to other users immediately. Any user can edit his/her local copy then forwards the updates of his/her local copy to other collaborators so that they can see the updates' effects immediately on their copies.
- **Asynchronous**: The enable users cooperation while updates may be observed with a delay at remote sites. Many tools support asynchronous communications such as Versions management tools like CVS [32] and file synchronizers like Unison [33]. For example, users can edit a shared file at different times using a file synchronizer. Then, their changes are merged later to get the same final view of the shared file.

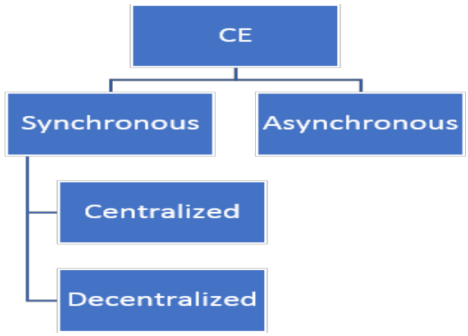

Fig. 2. Different kinds of collaborative editors.

In an opportunistic network, it should be possible to allow for both synchronous and asynchronous editing systems depending on the target application and the area hosting the opportunistic network. However, synchronous collaborative editors also known as Real-Time Collaborative Editors (RCE) are much more difficult to design and deploy over ON than asynchronous editors. Thus, we focus on RCE.

RCE are based on replicating the shared data to allow for high availability and to improve the performance of the editing system. Nevertheless, enforcing the data convergence (also known as consistency) of all replicas is hard to achieve. To solve this problem, many synchronization algorithms were proposed in the recent decades and can be classified into [30]:

- **Centralized algorithms**: This type requires the presence of a central server that is used to serialize concurrent users' updates as well as to get a unique and global order of operations execution ($e.g.$, GOT [34], COT [35], SOCT3 [36] and SOCT4 [36]).
- **Decentralized algorithms**: They allow requests that are concurrent to be executed in any order ($e.g.$,

SOCT2 [37], adOPTed [38], SDT [39], GOTO [40], ABTS [41] and OPTIC [6]).

Collaborative editing systems are consistent if they always maintain the following properties [40]:

- **Convergence**: All copies of the shared document converge towards the same state at different sites. This should be achieved if all sites execute the same set of updates even though the execution of updates happens in different orders.
- **Causality preservation**: If an update depends causally on another one, it should be executed after it at all collaborating sites.
- **Intention preservation**: For any operation, its execution effect at every site shall be the same as the intention of its first execution.

In the following, we discuss RCE properties and requirements.

A. Real-time Collaborative Editors (RCE)

RCE provide simple text editor user interfaces, allowing viewing and editing of the same document for a group of users from different sites simultaneously [42]. All modifications at each site are propagated and displayed at other sites. Therefore, the user at one site can see the remote user' modifications. REDUCE and Hydra are examples of RCE providing many features, $e.g.$, undo operations, lock certain sections of the text, variable granularity of text propagation and color highlighting of text (used to indicate text inserted by various user). RCE can be used for documentation and communication in many tasks including design and engineering [43]. RCE have two advantages [44]:

- Providing an environment for contribution of multiple users to shared documents in an easy and fast manner.
- Providing a platform for all users that is ready-to-use and does not need to install heavy software bundle $e.g.$, Libre Office or Microsoft Office. This platform enables users to view and modify their documents on their web browsers.

RCE should take into account human factors as follows [45]:

- **High local responsiveness**: The system shall be as responsive as a single-user editor.
- **Unconstrained interaction**: At any time, the users shall be able to edit any part of the shared document.
- **Real-time communication**: For effective coordination, the user must be aware immediately of each remote update.
- **Consistency**: The final version of the shared objects must converge ($i.e.$, be the same after the reception of all updates).
- **Scalability**: RCE must enable a group to be dynamic in order to allow users to enter or quit the group at any time [30].
- **Decentralized coordination**: To avoid a single point failure, all concurrent updates must be synchronized without relying to a central unit [30].

Since many users are allowed to edit the same object concurrently, divergence situations may occur which represents one of the most important challenges when designing RCE applications [46]. To maintain consistency while updating concurrently the copies of the shared document, the Operational Transformation (OT) approach was proposed [5]. It relies on an optimistic replication technique and synchronizes divergent replicas to produce a converged view of the shared document. Many collaborative applications use OT approach such as CoPowerPoint ($i.e.$, slides creation and presentation system of the real-time collaborative multimedia), CoWord ($i.e.$, a collaborative word processor) and Joint Emacs ($i.e.$, a groupware based on Emacs as text editor) [6]. Recently, an OT-based collaborative graphical editor named CoWebDraw has been proposed in [47].

Another alternative method for consistency maintenance is the Commutative Replicated Data Types (CRDT). It relies on commutative operations defined on abstract data types. To ensure convergence, a unique and globally ordered identifier is associated to every object [1], [48]. Various algorithms of CRDT have been proposed such as WOOT [49], WOOTO [50], WOOTH [51], RGA [52], Logoot [53], LogootSplit [54] and Treedoc [55]. Though CRDT proposes efficient solutions in term of time complexity, OT remains more used than CRDT in existing collaborative frameworks probably due to the less space complexity it requires. Thus, in this paper we focus on OT-based CE since there is no prior research works to adapt such famous collaborative editing approach over ON.

B. The Operational Transformation (OT) Approach

OT is a technique that achieves causality and convergence preservation while increasing responsiveness. It represents the most efficient and safest method for maintaining consistency. The aim of OT is to ensure the convergence of copies although the updates of users are executed out of order on different sites [6].

In OT approach, it is assumed that all updates are buffered at every site locally in a local log stored at each collaborating site. This allows remote updates to integrate the effect of concurrent updates over shared content composed by a sequence of elements ($e.g.$, a XML node, a page, a paragraph, etc.) [6]. Updates can be either local which are executed immediately or remote which need to be transformed before they are executed [56].

OT uses generally two primitive updates as follows [42]:

- **Ins**(p, e): is used to add the element e at the position p.
- **Del**(p): is used to remove the element at the position p.

To illustrate the principle of OT approach, let us consider the shared textual document initially containing the string $efecte$ and two updates $O_1 = Ins(2, f)$ and $O_2 = Del(6)$ performed by two different sites 1 and 2 concurrently, then the new states are $effecte$ and $efect$ respectively (see Fig. 3).

At site 2, when O_1 is executed, the state of the document is updated to $effect$. Meanwhile, at site 1, when O_2 is executed, it does not take into account O_1 that has been executed previously. Thus, the new state is $effece$. Consequently, the two sites have divergent states [6].

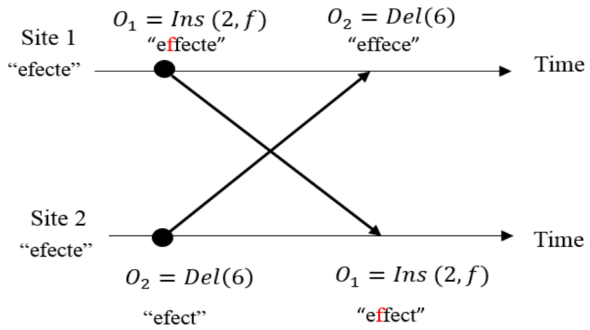

Fig. 3. Scenario of string operations without OT implementation.

To enforce convergence, OT relies on an algorithm called Inclusive Transformation (IT) [6], [30]. The effect of IT on the previous example leads to shift the deletion position of O_2 by 1 since the character f was inserted before O_2 is received at site 1. Thus, O_2 is transformed to $O'_2 = IT(del(6), Ins(2, f) = Del(7))$. The final resulting string at both sites is the same $effect$ and consistency is achieved as shown in Fig. 4.

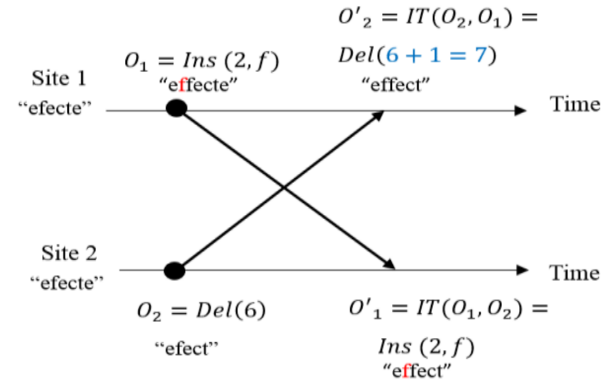

Fig. 4. Scenario of string operations with OT implementation.

OT defines another algorithm called Exclusive Transformation (ET) that allows to exclude the effect of an update from another one. ET is generally used to reorganize updates inside the log. Fig. 5 illustrates the ET function where O_1 is generated on the initial state abc and O_2 on the state produced by O_1, that is ac. Then $O'_2 = ET(O_2, O_1)$ means that ET transforms O_2 against O_1 to exclude O_1's effect. So, the result of O'_2 is $Ins(4, y)$ as if O_2 was generated on the initial state abc instead of the state produced by O_1 [6].

To ensure convergence, OT must verify two properties Transformation Property 1 (TP1) and Transformation Property 2 (TP2).

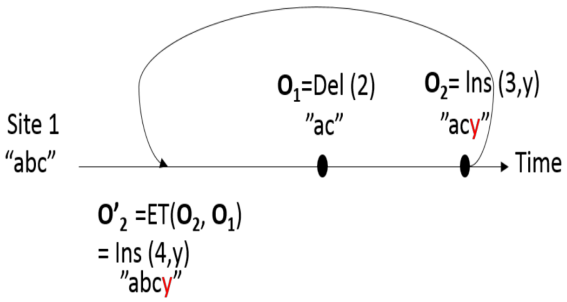

Fig. 5. Exclusive transformation example.

C. OT Properties

Two properties are required to achieve convergence using the OT approach [5], [36]:

1) **The first Transformation Property TP1**: This property was defined to ensure state identity, $i.e.$, if two sites begin the collaboration with the same initial state of the shared document, they must end with the same final state even if they execute the same updates but in different sequences. Formally, for any couple of sites having the same initial state S_0. If they perform concurrently the updates O_1 and O_2 then exchange their updates such that $O'_1 = IT(O_1, O_2)$ and $O'_2 = IT(O_2, O_1)$, it must be that $S'_1 = S'_2$ where S'_1 is the new state of site 1 after executing O_1 followed by O'_2 and S'_2 is the new state of site 2 after executing O_2 followed by O'_1.
2) **The second Transformation Property TP2**: This property defines updates identity and was defined to ensure that the transformation of any update let O_3 against equivalent update sequences ($i.e.$, two sequences including the same updates executed in different orders) must give the same result. It is formally defined as:

$$IT(IT(O_3, O_1), O'_2) = IT(IT(O_3, O_2), O'_1)$$

for any three concurrent updates O_1, O_2 and O_3 such that $O'_1 = IT(O_1, O_2)$ and $O'_2 = IT(O_2, O_1)$.

Fig. 6 presents an example that illustrates TP1, where there are two users sharing and editing the same document represented by a sequence of characters. At the beginning, the two copies contain the same string abc. At site 1, user 1 executes the update $O_1 = Ins(1, z)$ as a local update in order to insert the letter z at the position 1 and produces the string $zabc$. At the same time, user 2 performs $O_2 = Ins(2, y)$ as local update to insert the letter y at the position 2 and produces the string $aybc$ at site 2. At site 2, when O_1 is received and executed as a remote update, the string $zaybc$ is produced. At site 1, when O_2 is received and executed as a remote update, the new string $zaybc$ is produced. Obviously, the final string at sites 1 and 2 is consistent and TP1 is achieved.

Fig. 7 presents example of TP2, where there are three users working on a shared document with the same initial string abc. At site 1, user 1 executes the update $O_1 = Ins(3, x)$ as a

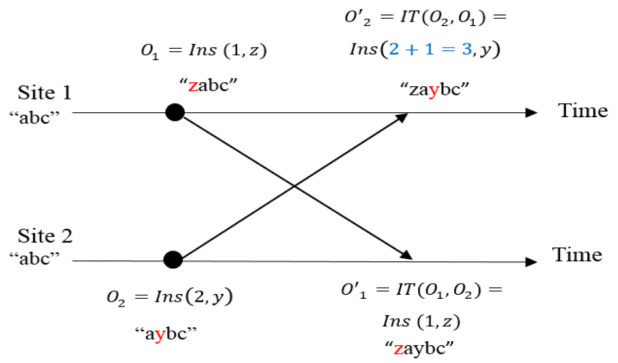

Fig. 6. TP1 example.

local update in order to insert the letter x at the position 3 and produces the string $abxc$. At site 2, user 2 executes the update $O_2 = Ins(1, z)$ as a local update in order to insert the letter z at the position 1 and produces the string $zabc$. At the same time, user 3 performs $O_3 = Ins(2, y)$ as a local update to insert the letter y at the position 2 and produces the string $aybc$ at site 3. TP2 is achieved at site 2 and site 3. In site 2, O_1 is transformed against O_2 and O'_3 to produce O'_1 on the state $zaybxc$. In site 3, O_1 is transformed against O_3 and O'_2 to produce O''_1 on the state $zaybxc$. Therefore, $O'_1 = O''_1$ and the final string at all sites is consistent.

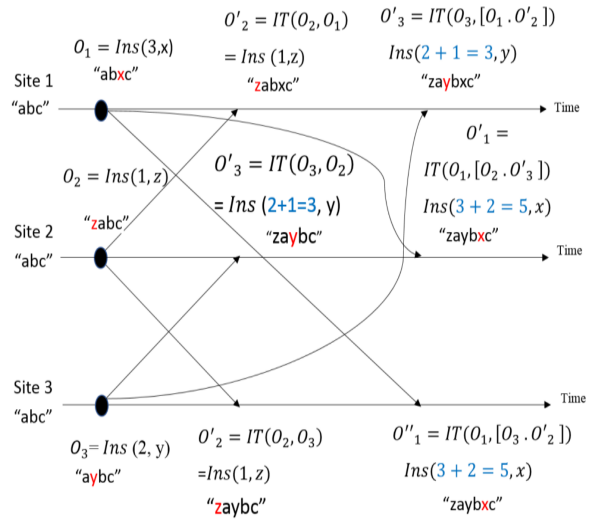

Fig. 7. TP2 example.

Though the OT principle is straightforward, many convergence problems may occur. Problems of OT approach will be presented in the following section.

D. OT Problems

There are three problems that can occur when applying OT approach in RCE:

First, the scalability issue consists in the ability of handling a high number of users efficiently. Indeed, dynamic groups must be enabled in RCE where the user can join and leave the group at any time. Vector timestamp technique has been used in most OT algorithms to determine the concurrent and happened before relations between updates. It consists of a finite vector of size n such that n is the number of collaborating sites. As a consequence, it does not scale well since vectors size is limited [6].

Secondly, the convergence is difficult to achieve. A killer scenario referred to as TP2 puzzle was discovered by Sun $et. al.$ [34]. Most existing OT algorithms fail to meet TP2 property which leads to data divergence situations. The TP2 puzzle occurs when two insertions and one deletion are generated concurrently as shown in Fig. 8. In this scenario, TP2 is violated at sites 2 and 3. In site 2, O_1 is transformed against O_2 and O'_3 to produce $O'_1 = Ins(2, x)$ on the state $axyc$. In site 3, O_1 is transformed against O_3 and O'_2 to produce $O''_1 = Ins(3, x)$ on the state $ayxc$. Therefore, $O'_1 \neq O''_1$ and this leads to data divergence [6].

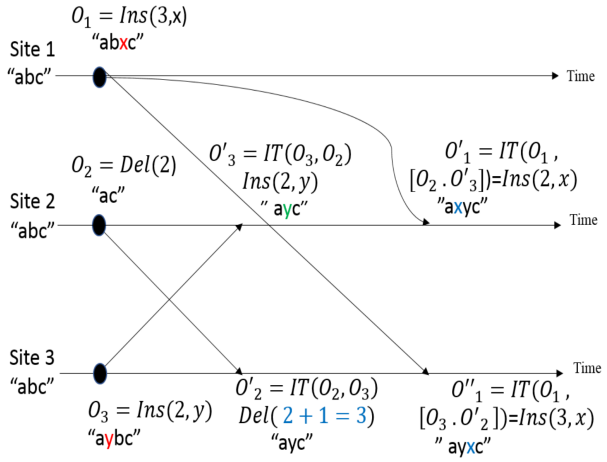

Fig. 8. Scenario of TP2 puzzle.

Lastly, the partial concurrency problem occurs when two or more causally dependent updates are generated concurrently to other updates. For example, if two sites begin the collaboration with the same initial state $fect$ and site 1 generates $O_1 = Ins(1, a)$ then $O_2 = Ins(2, f)$ to produce the $affect$ state while site 2 generates concurrently $O_3 = Ins(1, e)$ to produce the $efect$ state. Therefore, O_1 and O_3 are concurrent operations because O_1 none has seen the effect of the other while O_2 depends causally on O_1. Obviously, O_2 and O_3 are generated from different states which leads to partial concurrency situation. When O_2 is received as remote operation at site 2, it is transformed directly against O_3 and leads to divergence. Indeed, IT was defined for two concurrent operations while O_2 and O_3 are partially concurrent operations.

In the following, we compare the main OT-based CE algorithms according to a set of criteria that we have defined to meet the characteristics of ON.

IV. Comparing Existing OT-based Algorithms

Independently of the semantics of the shared objects, the integration algorithm proceeds in the following steps:

1) Generate the local update and execute it directly on the state of document.
2) Store this update in the log of the site.
3) Propagate the local update to remote sites.

The steps of applying the integration algorithm when an update is received from a remote site are [6]:

1) Search in the local log for all concurrent updates.
2) Use the transformation functions to integrate the effect of these concurrent updates.
3) Execute the transformed form of the remote update at the current document state.
4) Store the transformed form in the site's log.

Therefore, the integration algorithm builds the local logs while preserving the causal relation between updates of different copies of the document. In the following, we present the main OT-based integration algorithms that were proposed in the literature.

Ellis and Gibbs [5] have proposed the earliest OT algorithm; the distributed OPeration Transformation (dOPT). It is character-wise and offers good responsiveness. Moreover, it does not rely on a central server as well as fulfills the precedence property. However, dOPT enforces TP1 only, the dOPT-puzzle $i.e.$, a scenario leading to divergence, has been discovered [1], [57]. Thus, dOPT is unable to ensure copies convergence and does not solve the partial concurrency problem [36].

Ressel $et.\ al.$ have proposed the adOPTed algorithm as an improvement of dOPT [38]. It requires an additional transformation property TP2. To keep the track of all valid paths of operation transformations in adOPTed, a N-dimensional interaction model graph is used . The adOPTed algorithm solves both the partial concurrency problem [36] and TP2 puzzle [6].

Suleiman $et.\ al.$ have proposed SOCT2 [37] that uses both IT and ET functions. It solves TP2 puzzle [6] and the partial concurrency problem. However, it is very expensive and leads to performance degradation because it requires reorganizations of the local log many times to integrate each remote update [6].

The Generic Operation Transformation (GOT) concurrency control algorithm has been proposed by Sun et. al. [34]. It achieves convergence due to global total order. It defines an undo, do and redo scheme to integrate remote updates, thus avoiding TP2 puzzle [6], [58]. Indeed, out of order updates are first undone, then redone after the in-order updates are done. However, according to [1], GOT fails to provide correct results in some cases.

SOCT3 [36] uses a sequencer to maintain a global total order of execution via timestamps. The use of sequencer avoids TP2 puzzle by enforcing a continuous global order on updates [6], [58]. It depends on forward transposition and backward transposition (IT and ET). However, it does not scale well due to its client-server architecture [1], [6] and to the sequencer which represents also a single point of failure [39]. SOCT4 [36] is an improvement of SOCT3 that uses Forward Transposition (IT) only. Moreover, the partial concurrency problem has been solved [36].

Sun and Ellis [40] have proposed an optimization of GOT algorithm named GOT Optimized (GOTO). The new version ensures both TP1 and TP2. It has been tested in editing programs $e.g.$, CoMaya and CoWord. TP2 puzzle is solved [6]. It has been shown it solves the partial concurrency problem. Nevertheless, GOTO is very expensive and leads to performance degradation because it requires the reorganization of the local log many times to integrate each remote update [6].

Li and Li have proposed State Difference based Transformation (SDT) algorithm [39]. It is considered to be the first OT algorithm that is proved to converge in peer-to-peer group editors [57]. SDT has introduced the concept of update's effect relation and solves TP2 puzzle [6], [58].

Li and Li have also proposed the Admissibility-Based Transformation (ABT) [56] that does not require transformation functions to work. It depends instead on two correctness conditions that are formalized and proved: causality and admissibility preservation. The partial concurrency problem is solved by reorganizing the local log to integrate remote update [6].

Shao $et.\ al.$ [41] have extended the character-wise ABT algorithm to the Admissibility-Based Transformation with Strings (ABTS) algorithm which is string-wise. The correctness was formally proved and it has been shown ABTS does not need a total order and reversibility of updates to achieve convergence [41].

The Context-based OT (COT) algorithm [35] provides a framework for consistency maintenance and uniformed solutions for undo problems in distributed CE systems. Due to the nature of context vectors it uses, it may require extra memory space [59].

Imine [6] has proposed the OPerational Transformation with Intense Concurrency (OPTIC). OPTIC requires both IT and ET functions and introduces the semantic dependency for causality maintenance instead of state vectors. It scales naturally and is well-suited to manage highly dynamic groups. Moreover, OPTIC solves the TP2 puzzle using canonical logs ($i.e.$, special class of logs where insertions are stored before deletions). These logs enable transformation paths that lead to data convergence. Furthermore, OPTIC has a garbage collector mechanism [45], [60] to reduce appropriately log size without affecting the collaboration.

Based on the above discussion, we present an evaluation of the retained last version of most known CE algorithms in Table I. Our comparison is based on the following criteria that are of relevance in opportunistic context and allow for a correct deployment of CE over ON:

- **Correctness**: CE algorithms shall be correct to be deployed over ON. They shall ensure TP1 and TP2 and solve TP2 puzzle and partial concurrency problems.
- **Scalability**: To be easily deployed over ON, CE algorithms should be scalable to meet the high dynamic aspect of ON and be easy to use by an arbitrary number of users.
- **Decentralization**: Due to the high mobility and dynamic of ON, any proposed collaborative editing model dedicated to ON shall be completely decen-

tralized to take the benefits and keep the full potential of the opportunistic communication layer.

- **String handling support**: to reduce the communication overhead in ON, CE algorithms should be string wise. This is required to avoid the congestion problem.
- **Efficiency**: Since ON are constrained by the low storage and resources in peer nodes, the editing algorithm should not introduce additional memory and energy overhead. For this, we present in Table I the time and space complexities for each algorithm.

According to Table I, to achieve user intention preservation, all coordination algorithms use transformation of operations such as L-Transformation and multidimensional graph are used in adOPTed, Forward Transposition is used in SOCT4 and both IT and ET are used for other algorithms.

In order to achieve causality preservation, all the algorithms practically rely on state vectors except OPTIC which depends on semantic dependency relation.

To ensure copies convergence, algorithms adOPTed and GOTO refer to TP1 and TP2. SOCT4 refers to TP1 and continuous global order. SDT refers to IT, TP1 and TP2. ABTS uses admissibility preservation and causality preservation. Finally, OPTIC uses IT and ET algorithms, TP1, TP2, permutation function and canonical logs.

When it comes to the TP2 puzzle problem, all the algorithms presented in the table have succeeded to either solve or avoid the puzzle. Both ABTS and OPTIC use a special kind of logs called canonical logs in [6] that solves the TP2 puzzle.

As for the partial concurrency problem, GOTO solves this problem by reordering logs thus being expensive. The adOPTed algorithm solves the partial concurrency problem by constructing and memorizing a multidimensional graph in order to enable all the potential serialization orders to be retrieved. Also SOCT4 solves this problem thanks to deferred broadcast [36]. SDT solves this problem thanks to the effect relation between updates. ABTS uses admissibility preservation to ensure convergence and it has been formally proved. Lastly, OPTIC solves the partial concurrency problem thanks to the minimal dependency relation and avoids log reorganization before remote integrations.

Moreover, only OPTIC achieves the scalability property since it allows for unlimited number of users while the others are limited by the vector size.

Regarding to the decentralization criteria, all the algorithms are decentralized except SOCT4 since it relies on sequencer. Thus it is not well suited for ON.

The granularity of the shared object is very important. While the majority of proposed algorithms deal with characters, it is important for CE to allow for string handling in order to reduce the communication overhead over the network. As shown in Table I, all the algorithms support character except GOTO, ABTS and OPTIC since they can be easily extended to string elements.

Finally, the efficiency study shows all algorithms have linear time complexities except GOTO, SDT and ABTS that have a quadratic complexity, thus being inappropriate in ON. As for space complexity, adOPTed, GOTO, SDT and ABTS require huge memory space since they rely on state vectors. As a matter of fact, they all have complexity equal to $O(|H| * S)$ which is greater or equal to $O(S^2)$ if we assume each site generates at least one operation. The other algorithms have linear space complexity $O(|H|)$ and then are favored over ON mainly OPTIC since it provides a distributed garbage collector to clean logs [45], [60].

V. Overview on Collaborative Editing in Opportunistic Networks

In the last recent years, some research works have addressed CE over ON. In [7], a revision control system was adapted and developed over ON. The collaborative editing model was distributed and based on replicating shared documents. Each local copy contains the full history of previous versions. Two main approaches were proposed, namely adoption and merging in case a modified version of the content item is discovered. Adoption consists in adopting or discarding the modified version when two nodes carrying different versions meet opportunistically. Nodes receive modified versions from other peers then synchronize their copies as follows:

- The peer's version is ignored if it is a direct ancestor of the local version.
- The peer's version is adopted if it is a direct descendant of the local version.
- Otherwise, the merging is attempted.

Two decision making criteria are considered when selecting which version should be adopted or discarded: either the version presenting the latest changes is retained, or that providing the highest number of updates.

The merging approach is more complex and aims to generate a new version that takes into account all the updates performed on the shared data. A new version composed of the local version and the peer's version is produced or a conflict is issued. To proceed merge, the three-way merge approach, used in revision control systems, is used in order to automate merging. This approach, however, leads to conflicts when two versions cannot be combined. This occurs when users attempt to insert new content at the same position concurrently. To overcome the issue, it is required that the user interacts with the system to enforce convergence. Destructive and non-destructive conflict resolution are modeled. In the destructive model, only one set of modifications is selected while the other is discarded based on the version's length. Either the longer set among the two-conflicting change-sets is kept, or one of them is chosen arbitrarily but consistently if they have the same size.

In the non-destructive model, all the changes from both versions are retained by the node. Then, this node chooses to apply all changes from the first version then followed by the second version. This requires the application of these changes in the same order in all nodes to ensure consistency. An evaluation of merge and adoption was conducted to show that adoption performs well whereas merge outperforms adoption if the user intervenes in solving conflicts. There are many limitations in the proposed solution:

TABLE I. COMPARISON OF OT ALGORITHMS

Criteria/CE algorithms		adOPTed	GOTO	SOCT4	SDT	ABTS	OPTIC
Correctness	**Intention preservation**	L-Transformation and multidimensional graph [36]	IT and ET [36]	Forward Transposition (IT) [36]	IT and ET [39]	IT and ET [41]	IT and ET [6]
	Causality preservation	State vectors [36]	State vectors [36]	Timestamps [36]	State vectors [39]	State vectors [41], [61]	Semantic dependency relation [6]
	Convergence	TP1 and TP2 [36]	TP1 and TP2 [36]	TP1 and Continuous global order [36]	IT, TP1 and TP2 [39]	Admissibility preservation and Causality preservation [41]	IT and ET algorithms, TP1, TP2, permutation function and canonical logs [6]
	TP2 Puzzle	Solved [6]	Solved [6]	Avoided [6], [58]	Solved [6], [58]	Solved using canonical logs [41]	Solved using canonical logs [6]
	Partial Concurrency problem	Solved by constructing and memorizing a multidimensional graph in order to enable all the potential serialization orders to be retrieved [36]	Solved using reorder local log but it is very expensive and leads to performance degradation [6]	Solved thanks to deferred broadcast [36]	Solved thanks to the effect relation between updates [39]	Solved since characters order is respected in all sites [41]	Solved with minimal dependency relation and avoiding log reorganization [6]
Scalability		limited number of users	limited number of users	limited number of users	limited number of users	limited number of users	unlimited number of users [6]
Decentralization		Decentralized [30]	Decentralized [30]	Centralized [30]	Decentralized [30]	Decentralized [30]	Decentralized [6], [30]
String handling support		No [57]	Yes [57]	No [57]	No [57]	Yes [57]	Yes [6]
Efficiency[a]	**Time Complexity**[b]	$O(\|H\|)$ [58]	$O(\|H\|^2)$ [58]	$O(\|H\|)$	$O(\|H\|^2)$ [62]	$O(\|H\|^2)$ [41]	$O(\|H\|)$ [6]
	Space Complexity[c]	$O(\|H\| * S)$	$O(\|H\| * S)$	$O(\|H\|)$	$O(\|H\| * S)$	$O(\|H\| * S)$ [41]	$O(\|H\|)$

[a] To evaluate the efficiency, we present time and space complexities for each algorithm in the worst case.
[b] $|H|$ is the buffer (Log) size.
[c] S is the number of sites.

- The case where users cannot solve merge conflicts consistently is not considered such as when there are different users performing different merge strategies or when users change their selection over time.
- Authors have argued that the merging can be fully automated while it requires user interaction to solve merge conflicts.
- No correctness proof on the data convergence of the solution has been done.
- Authors have assumed the interaction with the user solves the conflicts while it may cause divergence.
- The solution leads to a blocking situation since the user can not modify the document because of the accumulation of unsolvable conflicts in the network.

Another collaborative editing solution over ON was proposed in [1]. It is based on CRDT and proposes OpportunisticLogoot as an adaptation of Logoot [53] to ON. It is similar to LogootSplit [54] in supporting single identifier for a sequence of characters to achieve total ordering. It has been shown that OpportunisticLogoot reduces the metadata sent with each message and the metadata size of document to be deployed over ON. Two operations are supported by OpportunisticLogoot for modifying the application's content and have the following syntax [1]:

- insert (pos, $base\_element$) to insert a $base\_element$ at position pos where $base\_element$ is the smallest element in the application's content like a character in a text editor or the whole message until the user presses Enter key in a chat.
- remove (pos) to remove a $base\_element$ stored at position pos.

The messages are ordered globally due to unique identifiers that are similar to identifiers in Logoot. These identifiers are generated in a densely ordered set and can be created between any two existing identifiers [1]. In order to ensure causality, the proposed algorithm can use external causal order algorithm like causal barriers [63] or vector clocks [64], [65]. OpportunisticLogoot algorithm uses a sequence of $base\_elements$ as content model instead of lines in Logoot. The definition of this sequence as pair $< id, sequence\_content >$ [1]:

- id is the unique identifier and is represented by a list of pairs $< x; s; l; r >$ and a logical clock $clks$, where x is a priority number used to sort the characters, s is the unique site identifier, l is the identifier of the first $base\_element$ in the sequence (or the range left limit), and r is the identifier of the last $base\_element$ in the sequence (or the range right limit).
- $sequence\_content$ is a group of $base\_elements$.

The identification system mentioned above ensures convergence without the need to transform updates. However, the proposed algorithm poses a series of limitations as follows:

- It relies on an identification system to ensure a total order between sequences. Though the authors have claimed the OpportunisticLogoot to reduce the memory space complexity, the solution still requires extra memory to store identifiers since for each sequence in the document a new identifier is created.
- It depends on existing causal order algorithms to ensure the causality.
- There are two kinds of messages, mutable and immutable. Thus, leading to some temporary blocking situations since it is not allowed to alter immutable messages.

Table II shows the comparison between the two aforementioned CE models over ON. We mainly discuss the testing settings over ON and compare the efficiency of the two solutions.

TABLE II. COMPARING CE OVER ON

Criteria	Shared Content [7]	OpportunisticLogoot [1]
CE Algorithm	Revision control mechanisms	Logoot [53]
Simulation Tool	ONE [18]	MobEmu [19]
Mobility (Trace or Model)	Modified Random Direction model (MRD) [21] and SMOOTH mobility model [22]	Infocom [24], Sigcomm [25] and UPB [26]
Routing Protocol	Epidemic [16] for MRD	ONSIDE [66] and Epidemic [16]
Efficiency	Time consuming because the whole history of the document is checked in each contact	Space consuming because it requires additional space for storing identifiers.
Convergence	There are blocking cases	There are blocking situations and the convergence is not evaluated

VI. CONCLUSION

Opportunistic Networks (ON) are emerging networks that are being more and more popular due to the large scale availability and use of mobile devices. In this paper, Collaborative Editors (CE) over ON have been addressed.

The major challenges that might arise when deploying CE over ON have been discussed including high mobility, dynamic of nodes and network delays. Moreover, the last versions of the most known OT-based CE algorithms were compared according to a relevant set of criteria. Finally, the current state of the art of CE over ON has been reviewed.

We believe that this study will be very useful in the area of CE over ON. As a matter of fact, this survey is intended to allow for a correct future deployment of CE over ON. It provides the main deployment constraints as well as the testing environments that might be useful to design and evaluate CE over ON. Furthermore, it allows to well define future research directions on how to run effectively CE over ON. It might be possible to either change CE algorithms or ON routing protocols for a better CE deployment over ON.

ACKNOWLEDGMENT

This research has been funded by the King Abdulaziz City for Science and Technology (KACST) under the reference 1-17-02-009-0015.

REFERENCES

[1] M. Costea, R. Ciobanu, R. Marin, and C. Dobre, "Causal and total order in opportunistic networks," in *Emerging Innovations in Wireless Networks and Broadband Technologies. IGI Global*, 2016, pp. 221–262.

[2] N. Kaur and G. Mathur, "Opportunistic networks: A review," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 2, pp. 20–26, Apr. 2016.

[3] J. Wu, S. Yang, and F. Dai, "Logarithmic store-carry-forward routing in mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 735–748, Jun. 2007, ISSN: 1045-9219.

[4] Z. Sui, "The research of the route protocols in opportunistic network," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec. 2015, pp. 192–196.

[5] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *1989 ACM SIGMOD international conference on Management of data*, Jun. 1989, 399407.

[6] A. Imine, "Coordination model for real-time collaborative editors," in *Field J., Vasconcelos V.T. (eds) Coordination Models and Languages. COORDINATION 2009*, 2009, pp. 225–246.

[7] T. Kärkkäinen and J. Ott, "Shared content editing in opportunistic networks," in *the 9th ACM MobiCom Workshop on Challenged Networks*, Sep. 2014, pp. 61–64.

[8] P. Asuquo, H. Cruickshank, Z. Sun, and G. Chandrasekaran, "Analysis of dos attacks in delay tolerant networks for emergency evacuation," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, Sep. 2015, pp. 228–233.

[9] J. Seedorf, A. Tagami, M. Arumaithurai, Y. Koizumi, N. B. Melazzi, D. Kutscher, K. Sugiyama, T. Hasegawa, T. Asami, K. K. Ramakrishnan, T. Yagyu, and I. Psaras, "The benefit of information centric networking for enabling communications in disaster scenarios," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–7.

[10] E. Y. A. Martin-Campillo J. Crowcroft and R. Marti, "Evaluating opportunistic networks in disaster scenarios," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 870–880, Mar. 2013.

[11] N. K. Kulesza, "User behaviour and security in opportunistic networks," in *Thesis, Department of Computing-Imperial College of London*, 2009, pp. 9–15.

[12] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson, and B. Plattner, "A decade of research in opportunistic networks: Challenges, relevance, and future directions," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 168–173, Jan. 2017.

[13] B. Poonguzharselvi and V. Vetriselvi, "Survey on routing algorithms in opportunistic networks," in *2013 International Conference on Computer Communication and Informatics*, Jan. 2013, pp. 1–5.

[14] A. R. Dinesh Singh Sanjeev Indora and A. Sharma, "Routing policies strategies in delay tolerant network," *International Journal of Engineering Research and Applications (IJERA)*, pp. 23–29, 2014.

[15] R. Rinky and R. Chauhan, "Routing protocols and movement models in opportunistic networks," vol. 01, pp. 69–72, Jan. 2016.

[16] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Tech. Rep., Apr. 2000, pp. 1–14.

[17] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," in *Service Assurance with Partial and Intermittent Resources: First International Workshop, SAPIR 2004, Fortaleza, Brazil, August 1-6, 2004. Proceedings*, P. Dini, P. Lorenz, and J. N. de Souza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 239–254.

[18] A. Keranen, "Opportunistic network environment simulator," in *Thesis, Helsinki University of Technology, Department of Communications and Networking*, Jan. 2008, pp. 6–25.

[19] R. I. Ciobanu, C. Dobre, and V. Cristea, "Social aspects to support opportunistic networks in an academic environment," in *Ad-hoc, Mobile, and Wireless Networks: 11th International Conference, ADHOC-NOW 2012, Belgrade, Serbia, July 9-11, 2012. Proceedings*, X.-Y. Li, S. Papavassiliou, and S. Ruehrup, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 69–82, ISBN: 978-3-642-31638-8.

[20] 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*, ser. Simutools '09, Rome, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, 55:1–55:10, ISBN: 978-963-9799-45-5.

[21] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser, "An analysis of the optimum node density for ad hoc mobile networks," in *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 3, 2001, 857–861 vol.3.

[22] A. Munjal, T. Camp, and W. C. Navidi, "Smooth: A simple way to model human mobility," in *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '11, Miami, Florida, USA: ACM, 2011, pp. 351–360, ISBN: 978-1-4503-0898-4.

[23] S. Batabyal and P. Bhaumik, "Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1679–1707, 2015, ISSN: 1553-877X.

[24] P. Hui and J. Crowcroft, "Bubble rap: Forwarding in small world dtns in ever decreasing circles," University of Cambridge, Technical Report UCAM-CL-TR-684, May 2007.

[25] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, "Mobiclique: Middleware for mobile social networking," in *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, ser. WOSN '09, Barcelona, Spain: ACM, 2009, pp. 49–54, ISBN: 978-1-60558-445-4.

[26] R. C. Marin, C. Dobre, and F. Xhafa, "Exploring predictability in mobile interaction," in *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, Sep. 2012, pp. 133–139.

[27] S. Noël and J.-M. Robert, "Empirical study on collaborative writing: What do co-authors do, use, and like?" *Comput. Supported Coop. Work*, vol. 13, no. 1, pp. 63–89, Jan. 2004, ISSN: 0925-9724.

[28] S. G. Tammaro, J. N. Mosier, N. C. Goodwin, and G. Spitz, "Collaborative writing is hard to support: A field study of collaborative writing," *Comput. Supported Coop. Work*, vol. 6, no. 1, pp. 19–51, Apr. 1997, ISSN: 0925-9724.

[29] S. Weiss, P. Urso, and P. Molli, "Compensation in Collaborative Editing," in *9th International Workshop on Collaborative Editing Systems - IWCES 2007*, ser. IEEE Distributed Systems Online, Sanibel Island, Florida, United States: IEEE, Nov. 2007, 6 p.

[30] A. Cherif, "Access control models for collaborative applications," in *Thesis, Nancy: Distributed, Parallel, and Cluster Computing [cs.DC]- University Nancy*, 2012, pp. 13–24.

[31] J. A. Preston and S. K. Prasad, "P2p document tree management in a real-time collaborative editing system," in *High Performance Computing – HiPC 2007: 14th International Conference, Goa, India, December 18-21, 2007. Proceedings*, S. Aluru, M. Parashar, R. Badrinath, and V. K. Prasanna, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 418–431, ISBN: 978-3-540-77220-0.

[32] P. Cederqvist, "Version management with cvs," *Network Theory Ltd*, 2002.

[33] B. C. Pierce and J. Vouillon, "What's in unison? a formal specification and reference implementation of a file synchronizer," Tech. Rep., 2004.

[34] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, "Achieving convergence, causality preservation and intention-preservation in real-time cooperative editing systems," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 5, no. 1, pp. 63–108, Mar. 1998.

[35] D. Sun and C. Sun, "Context-based operational transformation in distributed collaborative editing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 10, pp. 1454–1470, Oct. 2009, ISSN: 1045-9219.

[36] N. Vidot, M. Cart, J. Ferrié, and M. Suleiman, "Copies convergence in a distributed real-time collaborative environment," in *ACM conference on Computer supported cooperative work*, Dec. 2000, pp. 171–180.

[37] M. Suleiman, M. Cart, and J. Ferrié, "Concurrent operations in a distributed and mobile collaborative environment," in *Proceedings 14th International Conference on Data Engineering*, Feb. 1998, pp. 36–45.

[38] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhauser, "An integrating, transformation-oriented approach to concurrency control and undo in group editors," in *1996 ACM conference on Computer supported cooperative work*, Nov. 1996, pp. 288–297.

[39] D. Li and R. Li, "Ensuring content intention consistency in real-time group editors," in *24th International Conference on Distributed Computing Systems (ICDCS'04)*, Mar. 2004, pp. 748–755.

[40] C. Sun and C. Ellis, "Operational transformation in realtime group editors: Issues, algorithms, and achievements," in *1998 ACM conference on Computer supported cooperative work*, Nov. 1998, pp. 59–68.

[41] B. Shao, D. Li, and N. Gu, "Abts: A transformation-based consistency control algorithm for wide-area collaborative applications," in *2009 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Nov. 2009, pp. 1–10.

[42] D. Li and R. Li, "An approach to ensuring consistency in peer-to-peer real-time group editors," *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 17, no. 5, pp. 553–611, Dec. 2008.

[43] D. Chen and C. Sun, "Comparison of real-time text chat and collaborative editing systems," in *Cooperative Design, Visualization, and Engineering: First International Conference, CDVE 2004, Palma de Mallorca, Spain, September 19-22, 2004. Proceedings*, Y. Luo, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 196–204, ISBN: 978-3-540-30103-5.

[44] Q.-V. Dang and C.-L. Ignat, "Performance of real-time collaborative editors at large scale: User perspective," in *Internet of People Workshop, 2016 IFIP Networking Conference*, ser. Proceedings of 2016 IFIP Networking Conference, Networking 2016 and Workshops, Vienna, Austria, May 2016, pp. 548–553.

[45] M. D. Mechaoui, A. Cherif, A. Imine, and F. Bendella, "Log garbage collector-based real time collaborative editor for mobile devices," in *The 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2010, Chicago, IL, USA, 9-12 October 2010*, 2010, pp. 1–10.

[46] H. Song, Z. Ou, Y. Fu, X. Lin, and L. Yang, "Operational transformation algorithm with conflict detection and awareness methods for real-time collaborative editing in p2p environments," in *2011 IEEE International Conference on Computer Science and Automation Engineering*, vol. 2, Jun. 2011, pp. 571–576.

[47] L. Gao, D. Gao, N. Xiong, and C. Lee, "Cowebdraw: A real-time collaborative graphical editing system supporting multi-clients based on html5," *Multimedia Tools and Applications*, Nov. 2017, ISSN: 1573-7721.

[48] M. Ahmed-Nacer and P. Urso, "A framework for performance evaluation of decentralized eventual consistency algorithms," *EAI Endorsed Transactions on Collaborative Computing*, vol. 17, no. 11, Jun. 2017.

[49] G. Oster, P. Urso, P. Molli, and A. Imine, "Data consistency for p2p collaborative editing," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, ser. CSCW '06, Banff, Alberta, Canada: ACM, 2006, pp. 259–268, ISBN: 1-59593-249-6.

[50] S. Weiss, P. Urso, and P. Molli, "Wooki: a P2P Wiki-based Collaborative Writing Tool," INRIA, Research Report RR-6226, 2007.

[51] M. Ahmed-Nacer, C.-L. Ignat, G. Oster, H.-G. Roh, and P. Urso, "Evaluating crdts for real-time document editing," in *Proceedings of the 11th ACM Symposium on Document Engineering*, ser. DocEng '11, Mountain View, California, USA: ACM, 2011, pp. 103–112, ISBN: 978-1-4503-0863-2.

[52] H.-G. Roh, M. Jeon, J.-S. Kim, and J. Lee, "Replicated abstract data types: Building blocks for collaborative applications," *J. Parallel Distrib. Comput.*, vol. 71, no. 3, pp. 354–368, Mar. 2011, ISSN: 0743-7315.

[53] S. Weiss, P. Urso, and P. Molli, "Logoot: A p2p collaborative editing system," in *Research Report RR-6713*, Oct. 2008, pp. 1–13.

[54] L. André, S. Martin, G. Oster, and C. L. Ignat, "Supporting adaptable granularity of changes for massive-scale collaborative editing," in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Oct. 2013, pp. 50–59.

[55] N. Preguica, J. M. Marques, M. Shapiro, and M. Letia, "A commutative replicated data type for cooperative editing," in *2009 29th IEEE International Conference on Distributed Computing Systems*, Jun. 2009, pp. 395–403.

[56] D. Li and R. Li, "An admissibility-based operational transformation framework for collaborative editing systems," *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 19, no. 1, pp. 1–43, Feb. 2010.

[57] S. Kumawat and A. Khunteta, "Analysis of operational transformation algorithms," in *International Conference on Recent Cognizance in Wireless Communication and Image Processing. Springer, New Delhi*, Apr. 2016, pp. 9–20.

[58] R. Li, D. Li, and C. Sun, "A time interval based consistency control algorithm for interactive groupware applications," in *Proceedings. Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004.*, Jul. 2004, pp. 429–436.

[59] C. Hirsimaa and M. Nycander, "An analysis on operational transforms," in *Bachelors Thesis in Computer Science, School of Computer Science and Engineering-Royal Institute of Technology-Sweden*, 2010, pp. 12–14.

[60] M. D. Mechaoui, A. Imine, and F. Bendella, "Distributed log garbage collector-based real time collaborative editor for mobile and p2p environments," in *2011 11th Annual International Conference on New Technologies of Distributed Systems*, May 2011, pp. 1–8.

[61] M. Cart and J. Ferrié, "Asynchronous reconciliation based on operational transformation for p2p collaborative environments," in *2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007)*, Nov. 2007, pp. 127–138.

[62] D. Li and R. Li, "A performance study of group editing algorithms," in *12th International Conference on Parallel and Distributed Systems - (ICPADS'06)*, vol. 1, 2006, 8 pp.–.

[63] R. Prakash, M. Raynal, and M. Singhal, "An adaptive causal ordering algorithm suited to mobile computing environments," *Journal of Parallel and Distributed Computing*, vol. 41, no. 2, pp. 190–204, Mar. 1997.

[64] C. J. Fidge, "Timestamps in message-passing systems that preserve the partial ordering," in *the 11th Australian Computer Science Conference*, Feb. 1988, pp. 56–66.

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

[66] R. I. Ciobanu, R. C. Marin, C. Dobre, V. Cristea, and C. X. Mavromoustakis, "Onside: Socially-aware and interest-based dissemination in opportunistic networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–6.