Exploiting Content Centric Networking to develop topic-based, publish-subscribe MANET systems

Andrea Detti [*], Dimitri Tassetto [*], Nicola Blefari Melazzi [*], Francesco Fedi [†]

[*] CNIT University of Rome "Tor Vergata", Electronic Engineering Dept., {name.surname@uniroma2.it}

[†] Sistemi Software Integrati, R&D Dept.,francesco.fedi@ssi.it

Abstract— **Mobile Ad-hoc NETworks (MANETs) connect mobile wireless devices without an underlying communication infrastructure. Communications occur in a multi-hop fashion, using mobile devices as routers. Several MANET distributed applications require to exchange data (GPS position, messages, pictures, etc.) by using a topic-based publish-subscribe interaction. Participants of these applications can publish information items on a given topic (identified by a name) and can subscribe to a topic to receive the related published information. An efficient dissemination of publish-subscribe data in MANET environments demands for robust systems, able to face radio resource scarcity, network partitioning, frequent topology changes. Many MANET publish-subscribe systems have been proposed so far in the literature assuming an underlying TCP/IP network.**

In this paper, we discuss the benefits of building a MANET publish-subscribe system exploiting Content Centric Networking (CCN) technology, rather than TCP/IP. We show how CCN functionality, such as in-network caching and multicasting can be used to achieve an efficient and reliable data dissemination in MANET environments, including the support of delay tolerant delivery. We present different design approaches, describe our topic-based publish-subscribe CCN system, and report the results of a performance evaluation study carried out with real software in an emulated environment. The emulation environment is based on Linux virtual machines. The performance evaluation required also a CCN MANET routing engine, which we developed as a plug-in of the OLSR Linux daemon.

I. INTRODUCTION

The TCP/IP protocol suite is designed to *push* data to a remote host identified by an *address*. This host-centric service is not perfectly tailored for information-centric applications (e.g. web browsing) that require to *pull* information identified by *names*, regardless of the source of information. Users are interested in receiving "what" they requested, and do not care "who" provides it.

Nowadays, information-centric applications are the most used in the Internet, as well as in sensor and ad-hoc networks. The need of efficiently support them on top of a host-centric TCP/IP network has given rise to the introduction of many incompatible proprietary content-oriented services (name-based routing, caching, multicasting, data replication, and so on) offered, e.g., by Content Delivery Networks [1].

Information Centric Network (ICN) is an emerging paradigm aimed at supporting content-oriented services in any kind of network: wide and local area networks [5], mobile ad-hoc or mesh networks [15], delay tolerant

networks, sensor networks [35], etc. ICN rethinks network services by putting information dissemination at the center of the network-layer design. Rather than creating network pipes between hosts identified by addresses, ICN delivers information (or contents) identified by names. A user expresses an interest for a content to the ICN Application Programming Interface (API) and the underlying ICN functionality takes care of routing-by-name the content request towards the best source (be it the original one, a replica server, or an in-network cache) and of sending back the related data.

The ICN concept dates back to 1999, when the TRIAD architecture was proposed [3]. However, the interest in ICN has been rather limited until 2009, with few contributions from the research community. In 2009, V. Jacobson et al. presented an ICN architecture named Content-Centric Network (CCN) [4], aka Named Data Network (NDN), supported by an open-source implementation, named CCNx [7]. This work quickly renewed the interest in ICN research and most of current ICN work uses or is related to the CCN architecture.

CCN research mostly applies to wide area static network scenarios. However, CCN functionalities are deemed to be effective also in Mobile Ad-hoc NETworks (MANETs). In fact, many MANET applications require to exchange data (GPS position, messages, pictures, etc.) by using a publish-subscribe interaction scheme [9], which is information-centric in nature. Publishers characterize their information items with a set of attributes, and subscribers register their interests in receiving only those information items whose attribute match a given criterion, regardless of who is the publisher. Publishers and subscribers are decoupled in space (they do not need to know of each other), and in time (they do not need to be active at the same time). Such decoupling simplifies mobility and disconnected operations, which are typical of MANETs. Moreover, one-to-many delivery can exploit intrinsic broadcast properties of the wireless channel.

The simplest publish-subscribe scheme is the so called “topic-based” one, in which the only attribute of an information item is its being part of a given topic. Participants of a topic-based publish-subscribe system can publish information items on a topic and can subscribe to a topic to receive the related published information items. Topics are identified by a name (e.g. “foo.news”), and publishing an information on a topic *T* implies its distribution to the users subscribed to *T*.

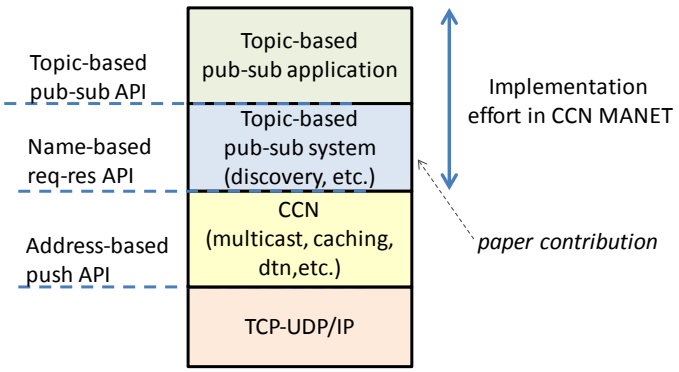

Fig. 1 – Functional architecture of a topic-based publish-subscribe system over CCN

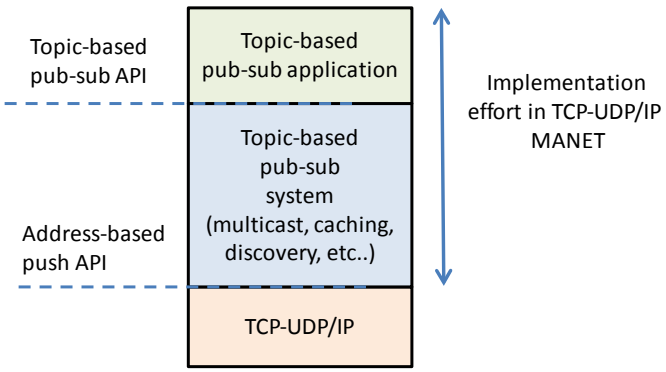

Fig. 2 – Functional architecture of a topic-based publish-subscribe system over TCP/IP

A more complex publish-subscribe scheme is the "content-based" one, in which publishers can characterize the information items with many attributes and subscribers can express complex conditions of interest. For instance a subscriber could be interested in information items regarding all cars with price less than 20k€. Content-based publish-subscribe systems are more complex than topic-based ones and their design is rather different.

The main contribution of this paper is to show how CCN functionality can be exploited to easily set up an efficient and resilient topic-based publish-subscribe system for MANETs. As shown in Fig. 1, we propose a topic-based publish-subscribe system that uses the service of a CCN. We use hierarchical names to address topic information items and this makes it possible to exploit the in-network caching and multicast functionality offered by CCN to achieve low delay and delivery efficiency. Moreover, through a dynamic configuration of CCN forwarding tables, we carry out a data muling functionality. Moving devices can use this functionality to transport information items among disconnected part of the networks, thus realizing the delay tolerant delivery [26] typical of Delay Tolerant Networks (DTNs).

Our findings are based on the practical development experience that we gained by using CCN as the underlying layer of the BEE DDS middleware [13], which is a specific implementation of the Object Management Group (OMG) specification of the Data Distribution Service for Real Time System [8]. We discuss design approaches, present some details of our pull-based system and report the results of a performance evaluation study carried out in an emulated environment, based on Linux virtual machines.

To evaluate the system performance, we had to develop our own CCN routing engine for MANET, since available CCN routing software tools, e.g. [6][34], are designed for fixed networks. The routing engine is developed as an open-source OLSR plug-in [23], which configures CCN routing tables according to a shortest path strategy. The plug-in may be of interest to other researchers, independently from the specific results

presented in this paper.

As for the organization of the paper, in section II we describe the CCN architecture and briefly revise some publish-subscribe literature works. In section III we motivate the use of CCN to realize publish-subscribe systems over MANETs and discuss the related research challenges. In section IV we discuss several approaches to build a publish-subscribe system over CCN. In section V we describe our publish-subscribe system. In section VII we report a performance evaluation study and in section VIII we draw our conclusions. Finally we present some additional performance results in an appendix.

II. Related Work

A. Information Centric Networking

Information Centric Networking (ICN) is a networking approach that addresses content by names, instead of locations, at the network layer, and combines in-network storage, multiparty communication and data centric security within a unique network framework aimed at providing efficient content distribution. So far several ICN architectures have been proposed. The paper [37] surveys several architectures that differ on how they implement ICN key functionality, such as:

- *Naming* – ICN names are location independent; different architectures adopt different naming schemes. Some of them, like CCN, use hierarchical names; whereas other ones (like PURSUIT or MobilityFirst) use flat names. This choice is usually related to the scalability of the underlying routing or resolution system.
- *Content routing* - To forward a content request towards a content source some architectures, like CCN or CONET [37], use on-path name-based routing tables and routing-by-name functionality; whereas other ones, like PURSUIT or MobilityFirst, use an off-path resolution system e.g. based on DHT.
- *Security* – ICN architecture embed security information in the content. Some architectures, like CCN, decouple security information from the names and this allows using human readable names; security information, e.g. signature, is inserted in the header of network data units and the system requires a PKI. Conversely, other architectures, like DONA or NetInf (or SAIL in [37]), use self-certified names that include an hash of the public key of the publisher in the name. These architectures do not requires a PKI, but names are not human readable.
- Communication primitives – Most architectures use a request-response communication primitive; an exception is PURSUIT that supports publish-subscribe communication primitives by means of a fixed network infrastructure formed by Rendezvous nodes organized through a DHT, and a global Topology

Manager.

We adopt the CCN solution and we briefly recall it in the next section.

B. Content Centric Network (CCN)

A CCN provides users with information items exposed by unique names. Information items (or contents) may be small data, chunks of files, etc.. A name that uniquely identifies an information item is formed by a hierarchy of strings, aka *components*, separated by a "/" character. The generic naming scheme is ccnx:/component#1/…/component#n. For the sake of simplicity we omit the default prefix "ccnx:/" in what follows.

To fetch an information item, a client sends an *Interest* message that includes the name of the information item. Then, the CCN network finds a source and sends back the information item to the client within a *Data* message (aka Content Object).

Fig. 3 shows the model of a CCN node. A name-based Forwarding Information Base (FIB) is used to *route-by-name* Interest messages using a prefix match strategy. A FIB entry contains a prefix and a list of *upstream* faces on which the Interest message may be forwarded to reach a source of information. A face is a generalization of the concept of interface and may be a connection to a next-hop CCN node or directly to an application party. For instance, in Fig. 3 Interests message for information items whose names contain the prefix "foo.news/P1" are forwarded on face 2. The FIB entries are configured through the CCN API, either manually or by a name-based routing protocol, not specified by the CCN architecture.

During the forwarding process of Interest messages, a CCN node leaves reverse path information <name – *downstream* faces> in a Pending Interest Table (PIT), where a downstream face is the face from which an Interest comes from. For instance, in Fig. 3 the node has received from the faces 0 and 1 the Interest messages for "foo.news/P1/2" and "foo.news/P1/3". When an Interest reaches a node having the requested information item (either in the cache or in a local application), the node sends back the information item within a Data message. The Data message is routed on the downstream path by consuming the reverse path information previously left in the PITs. If not consumed by a Data message, PIT entries expire after a time out period.

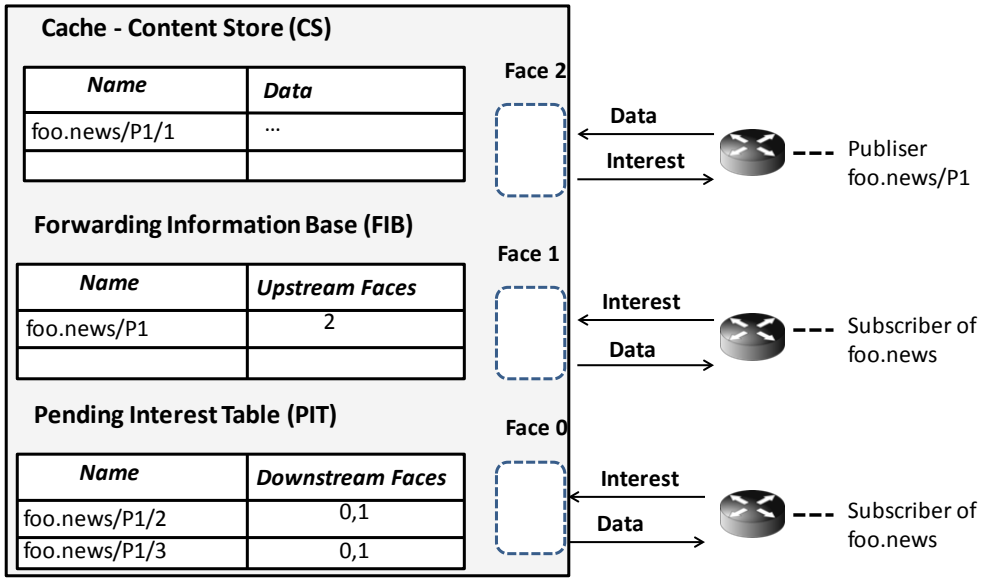

Fig. 3 – Model of CCN node

CCN nodes temporarily store the forwarded Data messages in a cache memory (e.g. foo.new/P1/1 in Fig. 3), named Content Store. The content store has a limited storing capacity and its use is controlled by a cache replacement policy, such as Least Recently Used (LRU), Least Frequently Used (LFU) or First Input First Output (FIFO).

If a node receives an Interest message related to a cached information item, the item is sent back immediately, without further forwarding the Interest message. Since caching occurs on all network nodes rather than only on edge nodes, this caching approach is usually referred to as *in-network caching*.

In case of concurrent Interest messages regarding the same information item, a CCN node forwards only the first received Interest message and stores in the PIT the set of downstream faces from which it received the next Interests. When the node receives the related Data message, it relays a copy of it towards all such downstream faces. In doing so a CCN node carries out a network-level *multicast* distribution.

CCNx [7] is a CCN implementation for Linux, Mac OS and Android platforms. The node model is implemented in C language, whereas the CCN API consists mainly of a wide set of Java libraries. The faces are, for the time being, UDP or TCP sockets. CCN functionality can be used either hop-by-hop (CCN deployed in each node of the network) or in an overlay fashion (CCN deployed in a subset of nodes connected by sockets).

C. Publish-subscribe systems

Publish-subscribe systems have been widely analyzed in the literature both for wired and wireless TCP/IP scenarios [16]. In case of wired TCP/IP networks, a traditional approach uses an always-on infrastructure of

message Brokers, which act as access points for publishers and subscribers, and store and forward information items from publishers to interested subscribers through an overlay distribution tree. Examples of such systems are [22][24].

In case of mobile ad-hoc networks these systems do not fit well, since they are not meant to support frequent topology reconfigurations caused by mobility. Since publish-subscribe is a form of multicast communication, from a publisher to many subscribers, many MANET approaches focus on the building and maintenance of a tree routing structure. For instance in [17], the authors studied the tree construction problem, proposed a related algorithm and analyzed it through simulation. Similar studies are [18][19][20][21]. Other structure-less MANET publish-subscribe systems provide information dissemination through optimized flooding or gossiping [32], since they consider too complex to maintain a distribution structure.

In the context of Delay Tolerant Networks (DTNs), publish-subscribe systems use nodes as *data mules* and take advantage of contact graph or of social relationships for routing purposes [33].

Whereas all the systems presented above use TCP/IP (or directly the MAC layer) as the underlying network service, some recent papers study publish-subscribe systems deployed on top of a CCN architecture. In [10][11] authors propose CCN publish-subscribe systems for fixed networks in which: i) subscribers set up a multicast distribution tree among CCN nodes by enhancing CCN with a name-based Subscription Table [11] or by using permanent entries in the PIT [10]; ii) publishers push information items on such a tree. To the best of our knowledge, this is the first paper that exploits CCN for publish-subscribe MANET systems.

III. Motivations and Challenges for using CCN in Publish-Subscribe MANET Systems

The advantages of using CCN and in general the information centric network paradigm in fixed networks have been thoroughly discussed [2]; in this section we motivate the use of this technology to realize topic-based publish-subscribe MANET systems, and discuss what we need to add to CCN for this purpose.

Why CCN for publish-subscribe systems in MANETs?

To ease the development of publish-subscribe applications, an application-developer usually exploits the API of an underlying publish-subscribe system or middleware, such as DDS [8], JEDI [22] or Java Message Service [24]. These systems provide developers with a publish-subscribe abstraction that hides the complexity of data dissemination. The resulting functional architecture is shown in Fig. 2: TCP-UDP/IP layers expose an address-based/push API (aka Socket) to the publish-subscribe system, which delivers information items from publishers to subscribers in an efficient and reliable way, and exposes to application designers a publish-subscribe API.

The realization of a publish-subscribe system on top of the TCP-UDP/IP API requires the development of a set of content-oriented functionalities, including: a discovery mechanism that identifies the IP addresses of topic publishers and/or subscribers, a protocol to express interest in specific information items (those of a topic) and retrieve them, a multicast distribution mechanism for efficient data dispatching, caching functionality, etc. In addition, the design of these functionalities is dependent on the targeted network environment.

Many literature papers discuss the design of publish-subscribe systems for MANETs (see [16] for a survey). However, most of them are studied only by means of simulations; the few papers reporting actual implementation do not make available the related software. In this context, the actual deployment of a topic-based publish-subscribe MANET application currently requires to develop both the application logic *and* the underlying publish-subscribe system (Fig. 2).

With respect to the publish-subscribe system development, we note that CCN natively offers most of the requested content-oriented functionalities, and its implementation is open source and well documented [7]. Therefore, we believe it is convenient to use CCN since it strongly reduces the effort of developing a topic-based publish-subscribe system. CCN provides multicast and in-network caching features which are very useful in critical network environments, such as in MANETs. Moreover, CCN mobile nodes may store-carry-and-forward missing information items to subscribers located in parts of the network temporarily disconnected by publishers, by using their caches. This opportunistic data muling is native in CCN and, as we will see, it only requires a simple configuration of the CCN routing table.

Why CCN by itself is not sufficient to support publish-subscribe applications?

Albeit the CCN content-oriented functionalities are of undeniable usefulness to publish-subscribe applications, the CCN API exposes them through a request (Interest) – response (Data) communication primitive, specifically designed for on-demand content delivery rather than for publish-subscribe data dissemination. Using the Interest-Data interaction provided by the CCN API, a user may request a "single" information item identified by name and the content is sent back if "already" published. Conversely, the subscriber of a topic-based publish-subscribe application demands for a "sequence" of information items identified by the topic name, and these information items will be published "after" the subscription time. Consequently, the request-response CCN API cannot be used by itself to support publish-subscribe applications and we have to introduce additional functionalities between the CCN layer and the publish-subscribe application, aimed at enhancing CCN with topic-based publish-subscribe functionality.

IV. DESIGN CHOICES

The development of a CCN topic-based publish-subscribe system requires specifying a naming scheme and a data dissemination model. This section describes the general solutions that we envisaged; in Section V we will present the proposed system as a whole and in details.

A. Naming scheme

Naming is the first design choice of a CCN-based system or application. A proper naming choice reduces the footprint on the routing plane and eases implementation through the CCN API.

A simple, nevertheless effective, naming scheme for information items of a topic-based publish-subscribe system is: topic-name/publisher-id/sequence-number. The topic-name component uniquely identifies a topic. The publisher-id component uniquely identifies the publisher of the topic (e.g. its MAC address or a random number). The sequence-number component is an incremental number that uniquely identifies the information items issued by a topic publisher. Overall, Tab. 1 reports the naming scheme that we consider.

Tab. 1 – Naming schemes

Description	**Scheme**	**Example**
Topic prefix	topic-name	foo.news
Publisher prefix	topic-name/publisher-id	foo.news/P1
Information item name	topic-name/publisher-identifier/sequence-number	foo.news/P1/1

B. Dissemination model

We envisage three different models to transfer information items from publishers to subscribers: *pull*, *push*, or *hybrid push/pull*. In this section we qualitatively describe these approaches, assuming the presence of a CCN routing protocol that distributes routing prefixes and properly configures the CCN FIB of the nodes.

1) Pull model

The Pull model can be deployed by using the plain CCN architecture and its characteristic are:

1- the CCN FIB of nodes contain entries related to publisher prefixes that are used to route Interests of information items toward relevant publishers;

2- subscribers *pull* the information items produced by publishers.

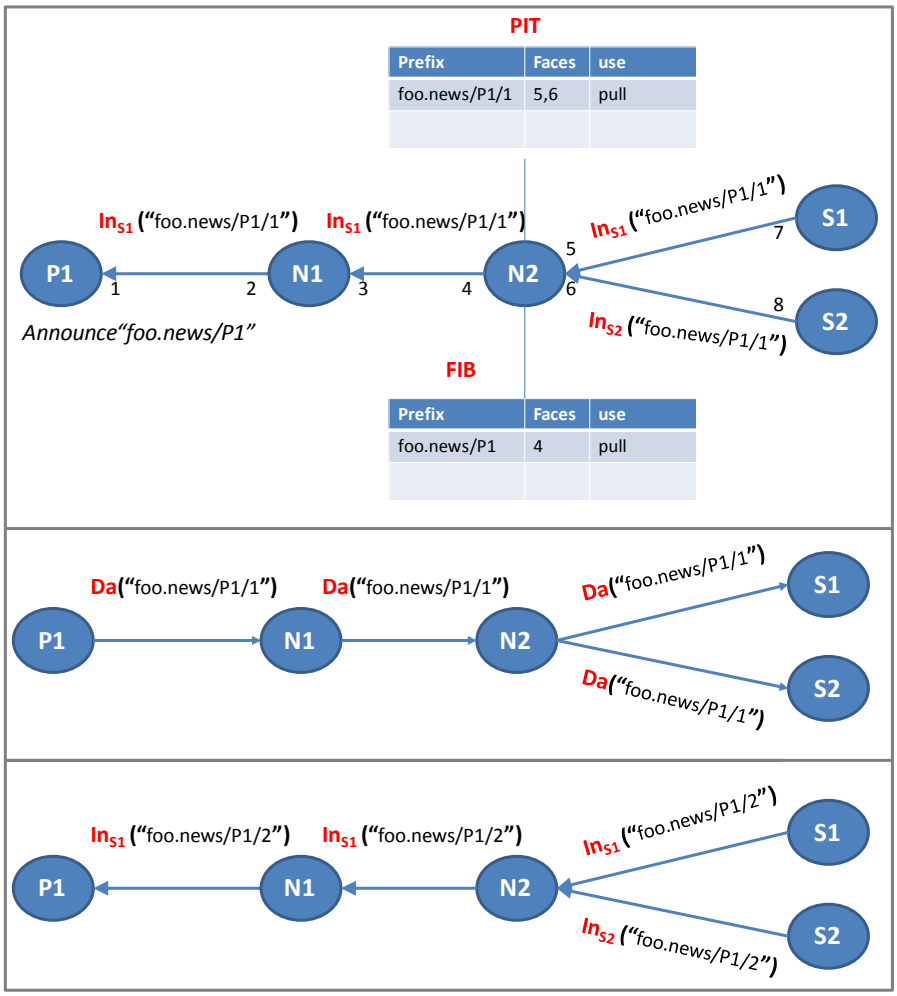

Fig. 4 – Pull model (In = Interest, Da = Data)

A CCN MANET routing protocol fills the FIBs of the nodes so that Interest messages containing the publisher prefix are forwarded towards the related publisher. For instance, in case of a proactive MANET routing protocol, a publisher may start the dissemination of routing announcements containing the publisher prefix, similarly to what happens in the case of BGP announcements of IP prefixes. This case is reported in Fig. 4, where the routing engine of publisher P1 announces the publisher prefix "foo.news/P1". The announcement is disseminated by the routing protocol, which inserts the publisher prefix in the FIBs of the network nodes. Thus, for instance, the FIB of node N2 is filled with the entry "foo.new/P1" that uses face 4 towards N1 as next-hop. In Section VI we report a CCN MANET routing protocol, which we devised to support our solution; nevertheless the pull model is independent of the CCN routing protocol.

To fetch information items, subscribers send requests (i.e. Interests) for the next information items that will be generated by the publisher and wait for the emission of the items. In Fig. 4 (top) subscribers S1 and S2 request the first information item "foo.news/P1/1". The network route-by-name Interest messages towards the publisher.

When the publisher produces the information item (Fig. 4 - middle), the network sends it back within a Data message, using the multicast dispatching tree built by the Interest messages. After the reception of the information item, the subscribers issue the request for the next information item by increasing by one the sequence number, then wait for it and so forth (see Fig. 4 - bottom).

It is to be noted that the PIT entry lifetime is limited to few seconds, so the multicast dispatching tree built by Interest messages has to be periodically renewed. This implies the use of a *polling* mechanism, and Interest messages have to be periodically resent, until the publisher generates the requested information item. Polling is an undeniable inefficiency, but, by re-sending Interest messages, subscribers refresh the topology of the multicast dispatching tree realized by the PITs; and this refresh operation is anyhow necessary in any MANET multicast distributing system, to cope with node mobility. In other words, the CCN MANET routing protocol is used to update the unicast routes from subscribers to publishers, while the Interest polling mechanism is exploited to update the multicast distribution trees from publishers to subscribers.

2) Push model

The characteristics of this dissemination model are:

1- the CCN FIB of nodes contain entries related to topic prefixes that are used to route information items of a topic (new CCN Push messages) towards the relevant subscribers;

2- publishers *push* the produced information items to the subscribers.

A CCN routing protocol used in the MANET fills the FIBs of the nodes so that information items whose name contain a topic prefix are forwarded towards the subscribers of the relevant topic. For instance, in case of a proactive MANET routing protocol, a subscriber may start the dissemination of routing announcements containing the topic prefix (e.g. “foo.news”) and the routing protocol uses these announcements to set up in the FIB a multicast dispatching tree from publisher to subscribers.

When a publisher generates an information item, the network uses the FIB entries to route-by-name the information items towards all subscribers. It is noteworthy that, differently from the pull model, in this case multicast dispatching uses the FIB rather than the PIT.

To support the push model, CCN requires the introduction of a new CCN message and of an appropriate multicast forwarding mechanism based on FIB information. The new data-unit, here named Push message, is equal to a Data message but is forwarded using the FIB, rather than the PIT. A Push message is relayed on each face that has a FIB prefix matching with the enclosed name and without storing any status in the PIT. This modus operandi resembles the IP multicast forwarding, with the difference that it is carried out using the name-based

CCN FIB, rather than the address-based IP FIB.

For instance, in Fig. 5 (top) we have two subscribers that announce the topic name "foo.news" on the CCN routing plane. Accordingly, a CCN routing protocol configures the FIBs of the nodes so that they forward the Push messages with prefix "foo.news" towards the announcing subscribers. The publisher P1 generates the first information item "foo.news/P1/1" and inserts it in a Push message. This message is forwarded (and in case cached) by intermediate nodes towards S1 and S2.

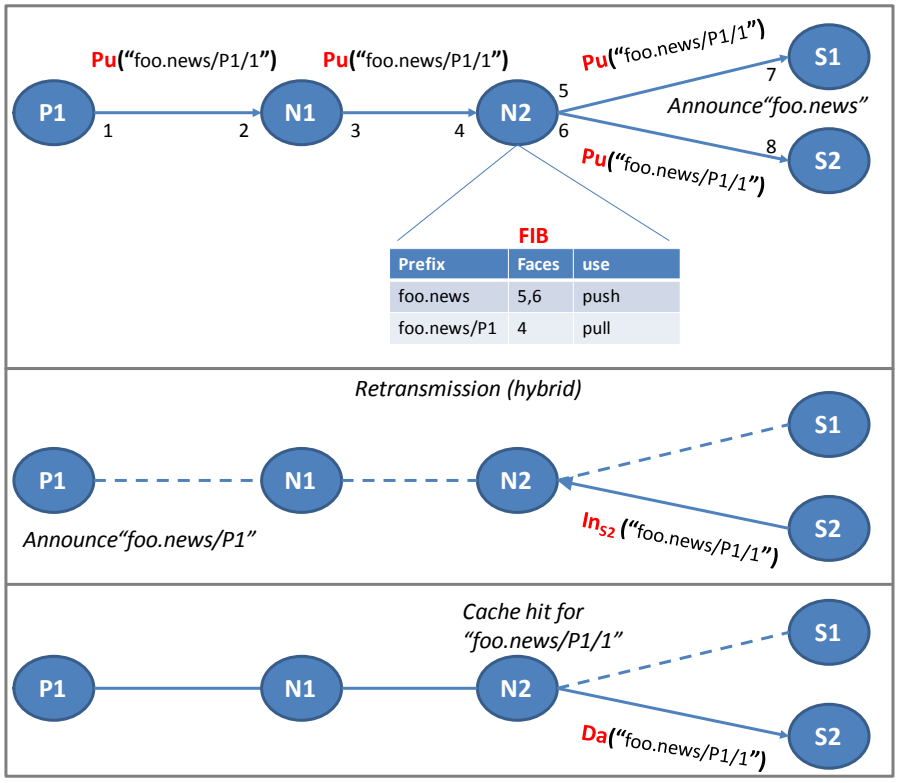

Fig. 5 – Push and hybrid approaches (In = Interest, Da = Data, Pu=Push)

C. Hybrid push/pull model

The Push model is theoretically more efficient than the Pull approach since it does not have to poll Interests, and do not use Interests at all. However, the Push approach lacks a fundamental aspect of publish-subscribe systems: the possibility of receiving information items also when publisher and subscriber are not connected at the production time or in case of data loss. This makes the Push model intrinsically *unreliable*.

To provide reliability, a Push/Pull hybrid model may be used, in which Pull is only invoked to retrieve missing items (e.g. inferred by out-of-delivery). In this case, the routing plane should handle routes toward both publishers (for Pull) and subscribers (for Push). For instance in Fig. 5 (middle and bottom), subscriber S2 lost the first transmitted Data "foo.news/P1/1" and requests its retransmission through an Interest-Data interaction with

the N2 cache.

V. TPS-CCN: OUR TOPIC-BASED PUBLISH-SUBSCRIBE CCN SYSTEM

In this section we present our proposal for a Topic-based Publish-Subscribe CCN system (TPS-CCN). TPS-CCN follows the naming scheme in Tab. 1 and it is based on the Pull model presented in section IV.B.1). TPS-CCN can provide *reliable* and *unreliable* delivery. In the reliable case, the system guarantees the delivery of any information items, whereas in the unreliable case the loss of information items may occur.

The protocol stack is shown in Fig. 1. CCN is used in every node; Interest and Data messages are routed-by-name hop-by-hop and thus CCN is the actual network layer of the system, rather than IP. Data transfers between neighboring nodes are supported by UDP/IP CCN faces. Clearly the presence of UDP/IP in the stack increases the stratification overhead and complexity. However, we point out that the UDP/IP layer is not strictly required by the TPS-CCN system since it only exploits the CCN API; but we used UDP/IP in our prototype because the current implementation of CCN is unable to operate directly on the MAC layer.

The TPS-CCN system is characterized by four procedures: Data Publish, Data Pull, Publisher Discovery and Sync, and Delay Tolerant Mode. These procedures are executed on top of the CCN application programming interface (API).

A. Data Publish

The Data Publish procedure is executed by a TPS-CCN publisher to make information items available to related TPS-CCN subscribers. As shown in Fig. 6, the TPS-CCN publisher is a CCN application connected to the underlying CCN node through a local face, e.g. 0xc. The CCN FIB has an entry related to the publisher prefix (e.g. “foo.news/P1”) that points to the local face. The publisher has an *Information Item Repository* (IIR) and a *Pending Interest Repository* (PIR). The data source stores each new information item in the IIR. In the case shown in Fig. 6 the source has published and stored in the IIR the information item “foo.news/P1/1”. Differently from the Content Store, the IIR is a storage module that is completely controlled by the publisher. Conversely, the publisher can not control what is stored by the Content Store since it is a cache for any forwarded Data, whose policy is internal to the CCN node.

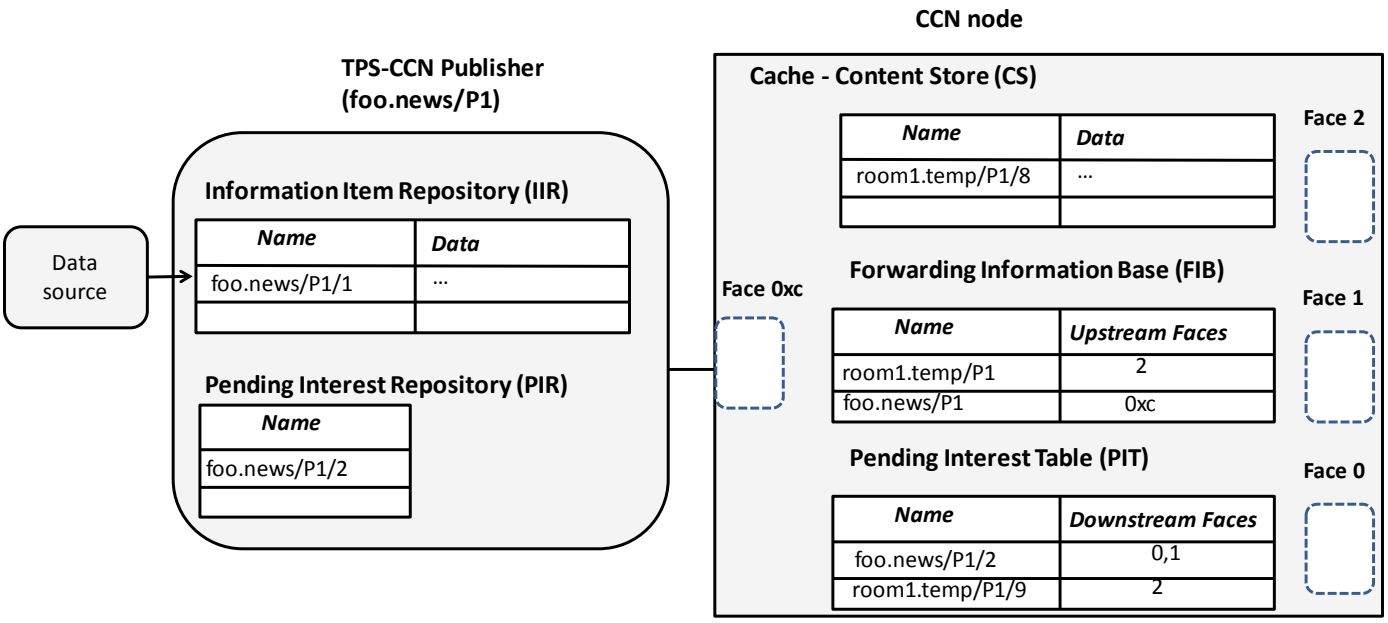

Fig. 6 – Model of CCN node with a TPS-CCN publisher

In case of an unreliable publish-subscribe system, the IIR only contains the last published information item. In case of a reliable system, the IIR contains all the information items that have been published since a period time not greater than a reliability timeout. In case the reliability timeout is configured greater than maximum publisher-subscriber disconnection period, then any published item is surely delivered. Clearly, a greater disconnection period requires a greater memory for the IIR. However, in MANET environments the disconnection period should be reasonable small (e.g. in the order of minutes) to make feasible the applicability of current memory technology.

When a node receives an Interest message for a local TPS-CCN publisher and the related Data message is not contained in the Content Store, the node uses the FIB to forward the Interest to the publisher application and also stores reverse route information in the PIT. When the TPS-CCN publisher receives an Interest message, if the requested information item is available in the IIR, it is immediately sent back to the underlying CCN node, which will forward it, consuming the related PIT entry. Conversely, if the item is not available, because it has not been produced yet, then the Interest is stored in the PIR. For instance, in Fig. 6, the publisher has received an Interest for the item "foo.news/P1/2" that is not yet published and this Interest is stored in the PIR (and also in the PIT).

The PIR is used to deliver an information item at the production time and without waiting for the next Interest polling from the subscriber. Indeed, when a new information item is produced, if there is a pending Interest in the PIR, then the information item is immediately sent back to the CCN node, which will use the PIT information to forward the item towards the subscribers.

We note that the information contained in the PIR is a subset of that contained in the underlying PIT. However, the CCN API does not provide access to the contents of the PIT; thus, without the PIR, the TPS-CCN publisher would be unaware of subscribers waiting for published information until the next Interest polling. The PIR entries expire after a time-out equal to the PIT one.

B. Data Pull

The Data Pull procedure is executed by a TPS-CCN subscriber to pull information items from the publishers. In case of multiple publishers for the same topic, a subscriber sets up a set of Data Pull procedures, one for each publisher. The main difference with respect to section IV.B.1) is the introduction of a *receiver window* that makes it possible to speed up data transfer in case of long round trip times. Through a polling mechanism, a subscriber continuously maintains in the network a window W of Interests, referring to the next W information items that the subscriber wishes to fetch from a publisher. If W is equal to 1, the interaction resembles to a receiver-driven stop-and-wait ARQ flow control mechanism, and thus the maximum throughput is one information item per round trip time. If W is greater than one, the maximum throughput is obviously W items per round trip time and the flow control mechanism resembles a receiver-driven Go-Back-N ARQ.

Fig. 7 reports an example of Interest Data flow between the subscriber S1 and the publisher P1. We consider a receiver window with size W=2. At the beginning of the interaction, the subscriber sends two Interest messages for items n.1 and n.2, whose names are "foo.news/P1/1" and "foo.news/P1/2" respectively. After a one-way transfer delay, the Interest messages reach the publisher, which has not yet published the related information items. At this moment, the subscriber has two Interests in the network for the next two missing items. After some time, P1 publishes the "foo.news/P1/1" information item and sends the related Data message. When S1 receives the Data message, it sends the Interest for the third item "foo.news/P1/3", so as maintaining in the network two Interests of the next two missing items ("foo.news/P1/2" and "foo.news/P1/3"). After a polling timeout the Interest of "foo.news/P1/2" is re-emitted by the subscriber. This polling timeout is equal to the timeout used by the network nodes to remove stale Interests from the PIT (aka Interest Lifetime). After a while, the publisher concurrently publishes the items "foo.news/P1/2" and "foo.news/P1/3". Since W=2, these two items have the related Interest pending in the network and can be immediately sent back within Data messages. When Data messages are received, the subscriber sends the Interest for next two missing segments, i.e. "foo.news/P1/4" and "foo.news/P1/5", and so forth.

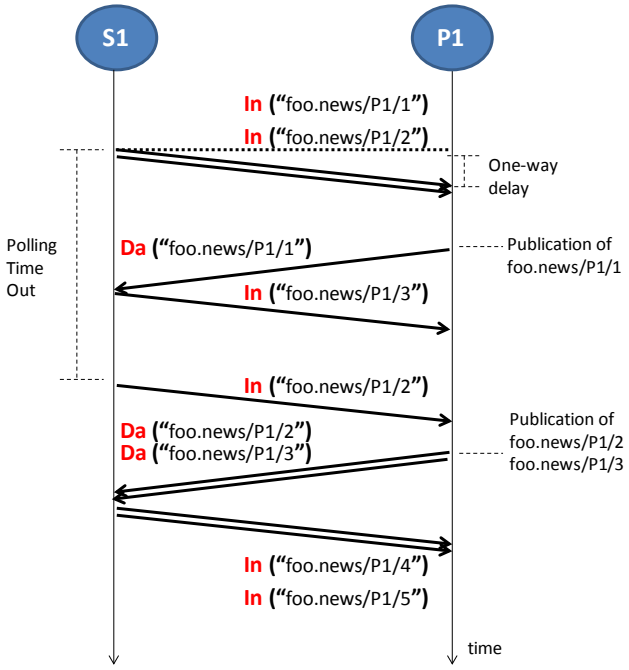

Fig. 7 –Flow control with *W*=2

C. Publishers Discovery and Sync

To pull information items from a publisher, a subscriber should know the name of published items, i.e. the components: i) topic name (e.g. "foo.news"), ii) publisher-id (e.g. P1) and iii) sequence number of the last produced information item. Assuming that the topic name is known a-priori, the subscriber periodically executes the Publisher Discovery and Sync procedure to retrieve or update the latter components.

The publisher-id component is derived by inspecting the CCN FIB entries that contain the topic name (e.g. "foo.news"). Indeed, the second component of these entries is the publisher-id. For instance in Fig. 4 the FIB contains the entry "foo.news/P1", thus P1 is the publisher-id of a publisher of the topic "foo.news".

After the discovery of the publisher-id component, a subscriber retrieves from the publisher its *publishing status*, i.e. the sequence number of the last published item and, in case of a reliable system, the sequence number of the oldest information item contained in the IIR. The publisher stores this control information in the IIR, by identifying it with a unique name, known a priori by the subscribers. The related naming scheme is: topic-name/publisher-id/status (e.g. "foo.news/P1/status"). A subscriber periodically fetches the publisher status by using the plain Interest-Data interaction. Since the publisher status is a dynamic data whose name instead does not change, to avoid caching inconsistency, the Data message that transports the publisher status is configured as not-cacheable.

D. Delay Tolerant Network mode

The Delay Tolerant Network (DTN) mode is a distinctive operational mode of TPS-CCN system that is implemented by managing the entries of the CCN FIB. The DTN mode enables subscribers to receive

information items even when they are in a part of the network disconnected from the publisher one. The DTN mode can be triggered when a subscriber becomes aware of being disconnected from the publisher. This can be inferred by the failure of a Publisher Discovery procedure.

We present the DTN mode with an example. As shown in Fig. 8 (top), in Normal mode (no DTN) the FIB of subscriber S2 has an entry for the publisher prefix "foo.news/P1" that uses face 8 (unicast UDP socket), connecting S2 to the next node N2 of the path toward the publisher. When this path disappears because of network fragmentation, face 8 is removed from the FIB by the routing protocol, and the next Publisher Discovery procedure fails. In this condition, the subscriber cannot fetch published information since the generated Interests do not match any FIB entry and are not forwarded. However, information items could be available in the cache of a neighbor node. To exploit this possibility, in case of a Publisher Discovery failure, a proximity FIB entry is added, with a face pre-configured as a multicast UDP socket. In this way, Interest messages are radio broadcasted to all neighbor nodes that send back the information item, if they have it in the cache.

As shown in Fig. 8 (middle and bottom), when S2 is in DTN mode it has a FIB entry for "foo.news/P1" pointing to face 10, which is a multicast socket. On this face, node S2 periodically sends the Interest message for the next expected item, i.e. "foo.news/P1/1". When S1 receives the Data from N2, it stores the message in its cache. Then, assuming that S1 moves towards S2 and S1 and S2 get connected, the Interest sent by S2 on the multicast interface reaches S1; then S1 sends back the cached Data. Thus, S2 receives the information item even if it is disconnected from the publisher. This delivery approach is also know in the literature as "data muling" (where S1 is the mule) or "store carry and forward", and it is the core of delay tolerant network forwarding mechanisms able to work in fragmented networks.

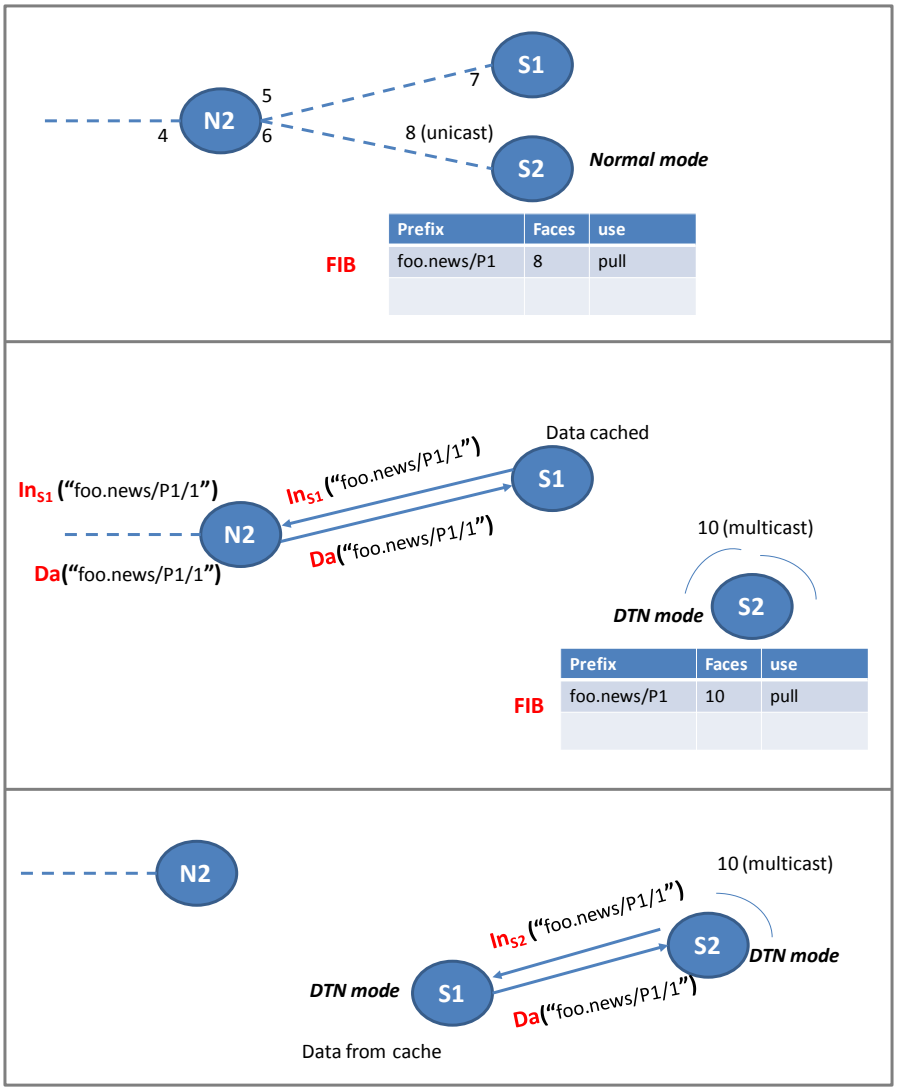

Fig. 8 – DTN mode (In = Interest, Da = Data)

VI. OLSR BASED CCN ROUTING ENGINE

TPS-CCN requires a CCN routing protocol. To the best of our knowledge, implementations of CCN routing protocols for MANET are not available, even if some analytical and simulation studies can be found e.g. in [29] [30]. Thus, to test the effectiveness of TPS-CCN we developed a MANET CCN routing protocol. Our protocol is based on the well-established OLSRd Linux daemon [28] of the OLSR protocol [27], enhanced by a new plug-in, named CCNInfo, which distributes the *location* of each name prefix, i.e. the IP address of the announcing node. Combining name prefix locations with the IP routing table provided by OLSR, a routing engine running on each node can readily determine the next-hop for each announced name prefix, i.e. the entries of the CCN FIB. As a result, the protocol sets up CCN multi-hop routes, following the shortest path identified by OLSR. This OLSR *evolutionary* approach may not be the best in terms of efficiency, but it is practical, simple and effective. The usefulness of the proposed routing tool (available in [23]) goes beyond the performance test of our TPS-CCN, as

it can be used in any other framework needing a MANET CCN routing mechanism.

The functional components of the routing tool are reported in Fig. 9. Each node has OLSRd and a routing-by-name engine module. A User (the TPS-CCN system in our case) announces a name prefix, communicating it to CCNinfo. The plug-in creates a CCNInfo OLSR announce message (see [23], readme file), which contains the tuple <nPrefix, destIP>, where nPrefix is the name prefix and destIP is the IP address of the announcing node. This message is flooded by OLSRd to all network nodes by using its default forwarding algorithm.

On each node, a routing-by-name engine module collects the received announcements in the nPrefix-to-destIP table, whose entries have a validity period; after that they are removed, if not refreshed. For each destIP, the routing engine derives which is the IP address of the next hop node by using the IP routing table provided by the OLSRd TxtInfo plug-in. The tuple <destIP, nextHopIP> is collected in the destIP-to-nextHopIP table, which resembles an IP routing table of entries with a /32 network mask.

Combining the information available in the two tables, nPrefix-to-destIP and destIP-to-nextHopIP, the routing engine computes which is the IP address of the next hop node for each announced name prefix (nPrefix-to-nextHopIP computation); then these relationships are pushed in the CCN FIB. To follow topology changes, the announcements are periodically resent, the destIP-to-nextHopIP table is periodically refreshed, and the CCN FIB computation is periodically repeated.

For instance, in Fig. 10 node P1 has IP address 192.168.0.3 and announces the name prefix “foo.news/P1”. Node S1 stores this information in the nPrefix-to-destIP. The destIP-to-nextHopIP table of node S1 indicates 192.168.0.2 (N1) as next hop node towards 192.168.0.3 (P1); thus, the CCN FIB is configured so that the upstream face for “foo.news/P1” is an UDP socket towards 192.168.0.2 (N1).

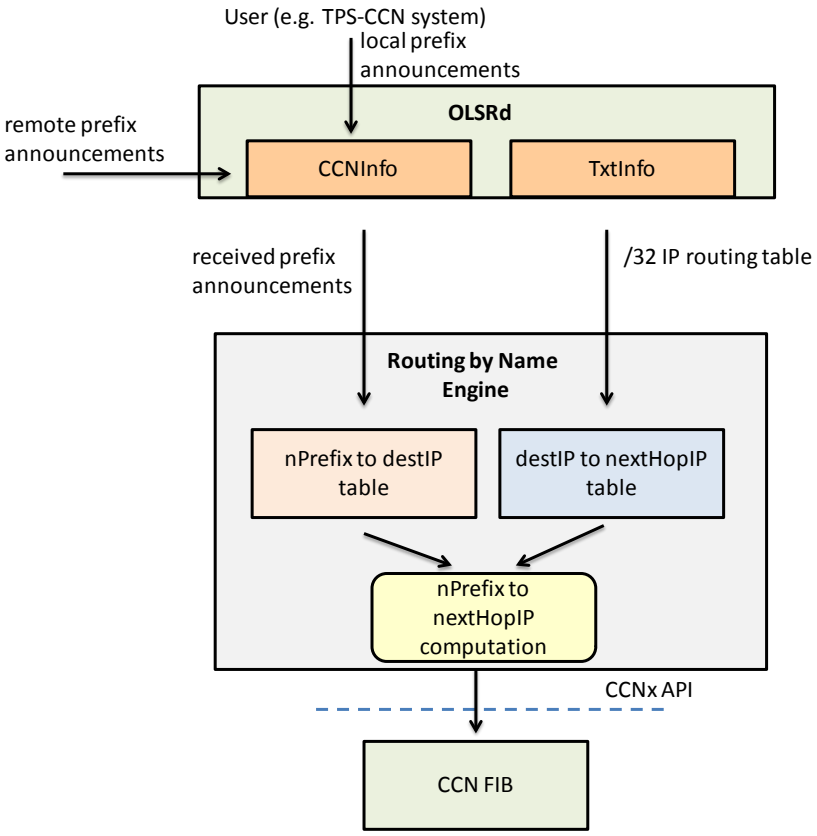

Fig. 9 – OLSR based CCN routing tool

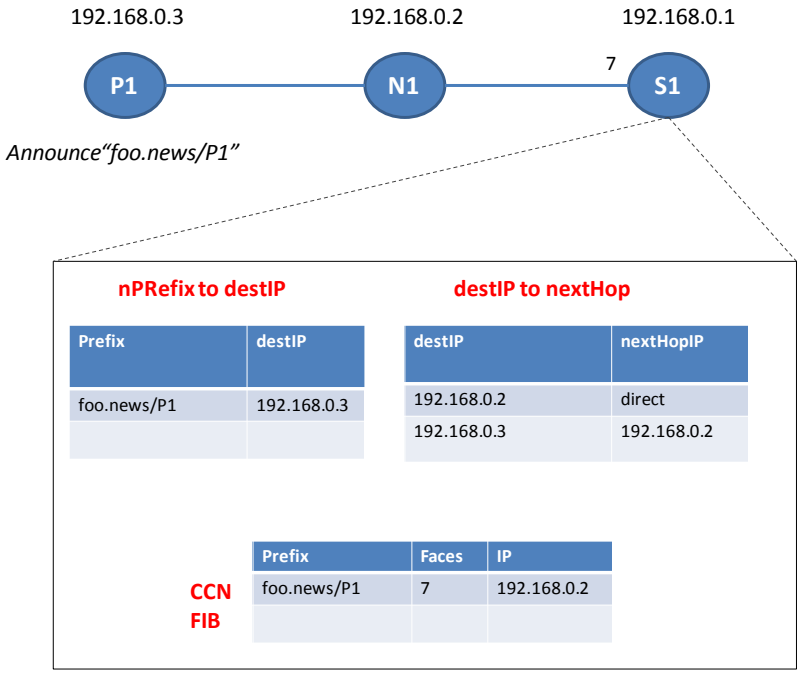

Fig. 10 –CCNInfo routing engine example

Tab. 2 – Emulation parameters

Parameter	Value
Scenario	
N. nodes	15
Mobility model	Random Way Point
Speed	2 m/s constant
Pause	5 s constant
Area	Up to 200m x 200m
Radio	802.11g at 54 Mbps constant
Tx Range	50m
N. Publishers / Topics	{1,2 or 3} with one publisher per Topic
N. Subscribers	N. nodes – N. Publishers, uniformly distributed on Topics
Traffic	one new information item every 1.1 seconds per publisher
Test duration	20 min

Parameter	Value
CCN and TPS-CCN	
CCN cache size	no-cache or 100 information-items
TPS-CCN Polling Time Out (PTO)	4 s
TPS-CCN Discovery Time Out	3 * PTO
TPS-CCN reliability timeout	5 min
TPS-CCN receiver window *W*	3
Routing	
Protocol type	OLSR (OLSRd daemon)
OLSR Hello Interval	1 s
OLSR TC Interval	3 s
OLSR TC Redundancy	2
OLSR MPR Coverage	7
OLSR CCNInfo plugin announce interval	5 s
OLSR CCNInfo plugin announce validity	200 s

VII. PERFORMANCE EVALUATION

In this section we report a performance evaluation of a Java implementation of the TPS-CCN system. We used the CCN implementation (aka CCNx) v0.7.2, OLSR with our CCNInfo plugin [23] and an emulation environment made up of Linux containers, whose setup has been controlled by the Common-Open-Research-Emulator (CORE) [14].

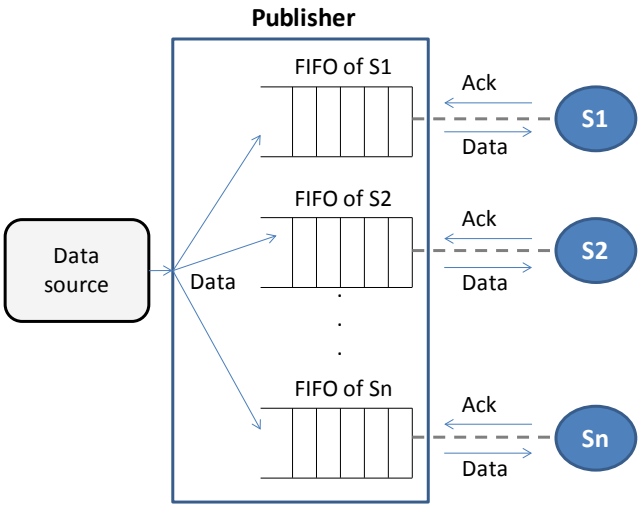

Fig. 11 – UDP-based publish-subscribe system

The aim of the performance evaluation is to understand the effectiveness of TPS-CCN functionalities taking into account MANET connectivity problems. To show the advantages brought about by data muling, in-network

caching and multicast of TPS-CCN, we also implemented a "basic" topic-based publish-subscribe system that only uses UDP sockets to transfer information items.

Fig. 11 shows the components of the UDP-based system. A publisher has a FIFO transmission queue for each subscriber. Each information item generated by the source is inserted in each queue. Each queue is controlled by an autonomous transfer process that sends queued items to the related subscriber through UDP unicast sockets. We implemented both a reliable and an unreliable UDP-based system. In the reliable case the first item of the queue is sent to the subscriber within a UDP datagram; when the item is received, the subscriber sends back an application level ACK message through an UDP message; the transmission of the item is repeated by the publisher each PTO sec until the ACK message is received, then the item is removed from the queue and the next queued item is transferred. In the unreliable case the first item of the queue is sent within a UDP datagram and immediately removed from the queue, independently of its reception.

We consider several scenarios whose configuration parameters are shown in Tab. 2. A possible use case of this scenario is a group of Unmanned Ground Vehicles (UGV) reconnoitering a critical area (e.g. in the aftermath of a disaster). The real-time computing complexity of our emulator limits to 15 the maximum number of nodes. Nevertheless, such a number is sufficient to understand the impact of TPS-CCN functionalities on performances, at least in a small scale MANET environment.

An emulation lasts for 20 minutes. During the first 18 minutes, the publishers generate a new information item every 1.1s. The next two minutes are used to allow the possible "delay-tolerant" delivery of last published information items through network movement. In doing so, we obtained the delivery of all published information items in case of reliable systems.

The measured performance parameters are: the mean delay between publishing time and receiving time, also known as delivery delay; the ratio between the number of received and sent information items, also known as delivery ratio; and the number of packets transmitted by network nodes (including OLSR routing data), normalized to the number of delivered packets. The latter is a measure of the system efficiency in delivering information items.

At a first glance the area of the scenario (up to 200m x 200m) may appear rather small if compared with the radio coverage (50 m). However, we measured a significant difference between *radio connectivity* and *network connectivity* (i.e. two nodes are connected if the routing protocol can set up a radio path between them). It is the network connectivity that impacts application performance. Fig. 12 reports the probability of having network

connectivity between two random nodes. We consider different mobility traces, for different lengths of the side of the scenario area and different numbers of nodes. To measure the network connectivity, each node randomly extracts a destination per second and sends an IP PING to it; if the PING is successful there is network connectivity. For an area of 200m x 200m the network connectivity is quite low because the routing protocol is not able to quickly reconfigure network paths, even though we used OLSRd configuration parameters deemed to fit MANET scenarios.

Given these results we consider it meaningless to investigate the performance of TPS-CCN for areas greater than 200m x 200m, since *TPS-CCN mechanisms are meant for MANETs scenarios in which the presence of an end-to-end path is not a very rare event* (the DTN mode helps in case of partitioning but should not be the main mode of operation). Extremely sparse MANET scenarios would require procedures not included in TPS-CCN, such as controlled data replication [25] (and not only the simple en-route caching), DTN routing rather than plain OLSR [31], monitoring of node contacts, etc.

In what follows we present some of the results obtained for reliable and unreliable systems. Additional results are reported in appendix I.

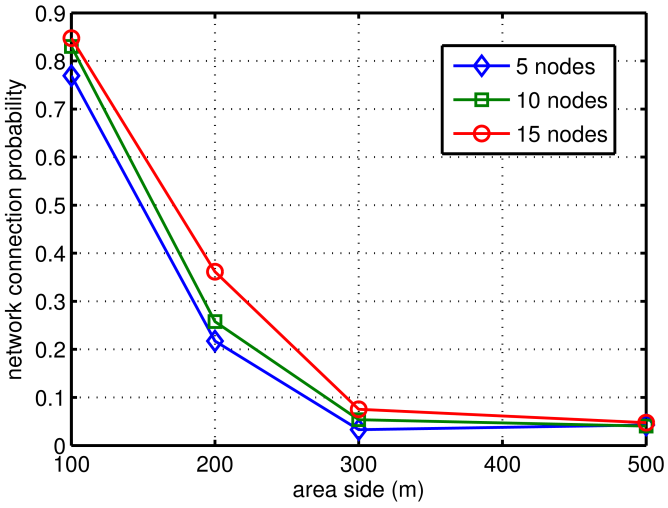

Fig. 12 – Network connection probability

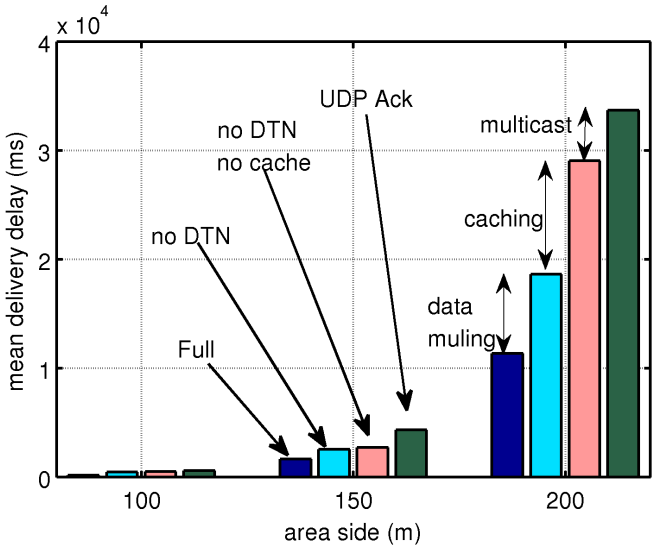

Fig. 13 – Mean delivery delay, reliable mode, 2 topics, 15 nodes

A. Reliable publish-subscribe system

In this section we focus on a reliable topic-based publish-subscribe system, in which all published information items are, sooner or later, delivered in order to subscribers. To understand the effect of the TPS-CCN and CCN functionalities we consider:

- a TPS-CCN "Full" configuration, with data muling (i.e. DTN mode enabled), in-network caching

(provided by CCN level caches) and multicast distribution functionalities (provided by CCN PIT mechanisms);

- a TPS-CCN "no-DTN" configuration, with in-network caching and multicast distribution functionalities;
- a TPS-CCN "no-DTN/no-cache" configuration, with only multicast distribution functionalities;
- the "UDP-Acked" publish-subscribe system, without any of the above functionalities.

Fig. 13 reports the delivery delay in case of 2 topics versus the length of the area side. The increase of the area side leads to a valuable grow of the delivery delays due the increasing occurrence of network partitioning events, which may disconnect publishers from subscribers for many seconds.

The performance difference between the Full and no-DTN configurations is due to the exploitation of data muling. When DTN mode is disabled (no-DTN) subscribers located in parts of the network disconnected from the publisher do not receive information items, even though the items could be available on the CCN cache of close nodes. Disconnected subscribers should wait that mobility creates an end-to-end path with the publisher to receive missing items, and this waiting time increases the delay with respect to the Full case, in which subscribers can fetch data from neighboring caches. We remind that disconnection periods are due not only to an actual absence of a radio path, but also to reconfiguration lags of the routing protocol. For instance, we measured that when two nodes get in radio coverage, then OLSRd spends about 3 seconds to recognize this connectivity condition. In the 200m x 200m case, disconnection periods occur frequently, and for both radio and routing issues. By decreasing the area, the duration of disconnection periods decreases as well and routing reconfiguration lags tend to be the principal disconnection cause. Obviously, the shortening of disconnection periods reduces the effectiveness of data muling as confirmed by the results of Fig. 13, in which the Full vs. no-DTN performance gap tends to vanish as the scenario area decreases.

The performance difference between the no-DTN and the no-DTN/no-cache configurations is due to the exploitation of CCN caches. The presence of in-network caches reduce the length of the network path traversed by an information item to reach a subscriber, with a consequent reduction of the delivery delay. The advantage of network path reduction increases with the area of the scenario. In the 200m x 200m case the network path reduction is valuable and the lack of CCN caches causes a considerable increase of the delivery delay. For smaller area side, network paths tend to shorten and caches produce a small reduction of the path length; thus, the no-DTN and no-DTN/no-cache configurations tend to show similar performance.

The performance difference between the no-DTN/no-cache and UDP-Ack is due to the network multicast

distribution provided by CCN PIT. The greater the network paths (as in the 200m x 200m case), the greater is the performance benefit.

We can conclude that *TPS-CCN provides better delay control for large areas, in which path lengths and disconnection periods are relevant*. Clearly, delivery delays may be high in case of fairly disconnected scenarios (e.g. 200m x 200m), since we are considering a reliable publish-subscribe system and a long time may elapse before a subscriber gets connected with the publisher or finds the data in a neighbor cache (only for Full configuration).

Fig. 14 reports the mean delivery delay versus the number of topics, in case of 15 nodes, for the 200m x 200m scenario. In the UDP-Ack case, performance is rather independent from the number of topics, since this solution is based on point-to-point transmissions. Conversely, in the other three TPS-CCN configurations, there is a slight performance degradation at the increase of the number of topics. This is motivated by the smaller effectiveness of CCN caching and multicasting at the decrease of the number of subscribers per-topic (see Tab. 2), i.e. of the number of users interested to the same information item.

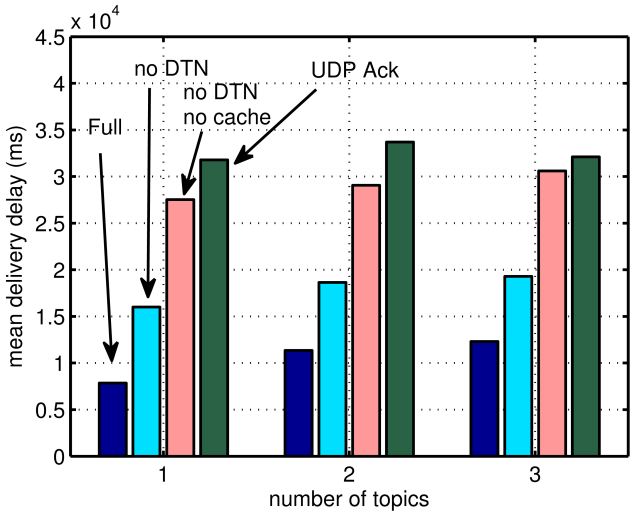

Fig. 14 – Mean delivery delay, reliable mode, 200m x 200m, 15 nodes

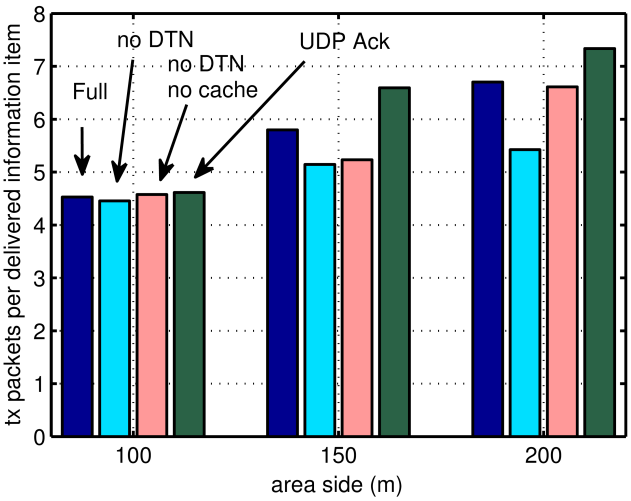

Fig. 15 – Number of transmitted packets per delivered information item, reliable mode, 2 topics, 15 nodes

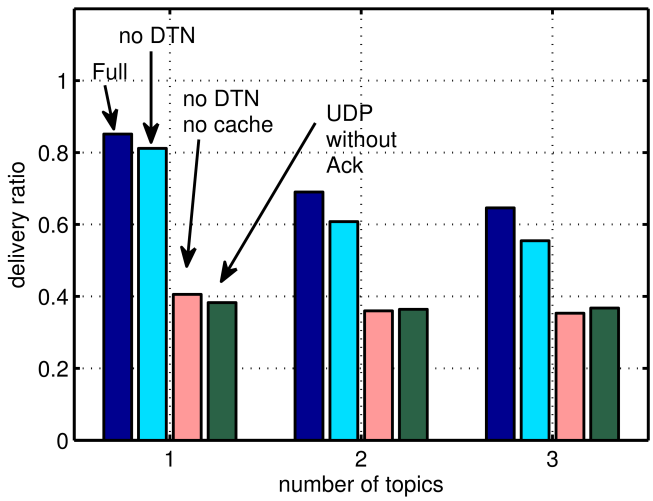

Fig. 16 – Delivery ratio, unreliable mode, 200m x 200m, 15 nodes

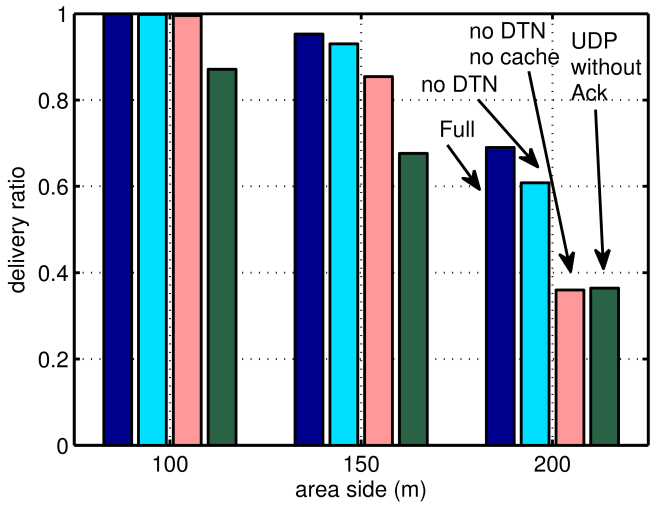

Fig. 17 – Delivery ratio, unreliable mode, 2 topics, 15 nodes

By lowering the number of subscribers, the distribution of an information item involves a lower number of network nodes and a lower number of in-network caches. Such a lower replication of information items decreases both the probability of finding the item in a cache and the number of data mules. In addition, by lowering the number of subscribers the effectiveness of a multicast distribution with respect to a unicast one (e.g. the UDP-Ack) tends to vanish. Overall, we can conclude that *the effectiveness of the TPS-CCN functionalities increases with the scale of the distribution*.

Fig. 15 reports the number of packets transmitted by network nodes (including OLSR data), normalized to the number of delivered packets vs the area size. Performance differences show up at the increase of the area. With respect to the no-DTN configuration, the Full one may consume more network packets since, when a subscriber is disconnected from the publisher, it sends out Interest messages to probe the presence of information items in neighboring caches; this does not happen in the no-DTN configuration. With respect to the no-DTN/no-cache case, in large areas the DTN overhead tends to be compensated by the inefficiencies due to the absence of in-network caches.

B. Unreliable publish-subscribe system

In this section we analyze an unreliable publish-subscribe system. Again, we consider four configurations: Full, no-DTN, no-DTN/no-cache, and UDP without Ack.

Fig. 16 and Fig. 17 show the delivery ratio in an area of 200m x 200m versus the number of topics, and in case of 2 topics versus the length of area side, respectively. The delivery ratio is the rate between the overall number of information items received by subscribers and the overall number of items that the subscribers would have

received in absence of loss (e.g. due to the temporary absence of network connectivity). The difference between the Full and the no-DTN cases is rather limited and this implies that the impact of data muling on delivery ratio is limited too. Conversely, the difference between the no-DTN case and the no-DTN/no-cache case is valuable, especially for large areas. Therefore *CCN caches have a relevant impact on delivery ratio*. Without cache, the delivery ratio performance resembles the one of the simple UDP system.

Fig. 16 shows that the performance of CCN configurations decreases at the increase of the number of topics. As we already discussed for the reliable system, this is due to a decrease of the TPS-CCN effectiveness at the reduction of the number of subscribers per topic. Moreover, the performance differences reported in Fig. 17 indicate (again) that the effectiveness of TPS-CCN increases with the area side.

Fig. 18 reports the mean delivery delay in case of an area of 200m x 200m versus the number of topics. Fig. 19 reports the delivery delay in case of 2 topics versus the length of area side. These results could be misunderstood if we do not take into account that delays are evaluated only on delivered information items, and that different configurations provide a different number of deliveries (Fig. 16), especially for large areas. Exploiting the CCN caches available in the Full and no-DTN configurations, the subscribers are able to gather missing information items even several seconds after the emission time. Therefore, the delivery ratio is high but delivery delays may be very long and this increases the mean delay value.

However, if delays are not suitable for the application, CCN makes it possible to configure the maximum storing time of an information item in a cache. In absence of caches (no-DTN/no-cache and UDP) either the information item is successfully delivered when emitted or it is lost. Thus, the number of delivered information items is low, but all delays are short. In some configurations, the delay tends to slightly decrease as the number of topics increases. This is related to the decrease of the delivery ratio: indeed a lower number of aged items are delivered.

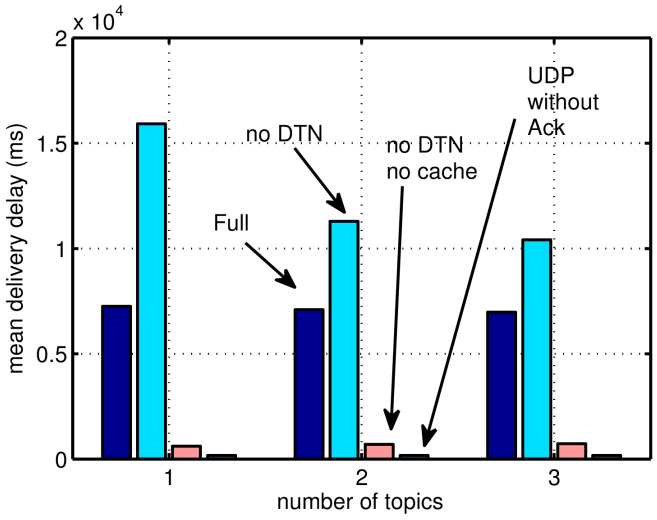

Fig. 18 – Mean delivery delay, unreliable mode, 200m x 200m, 15 nodes

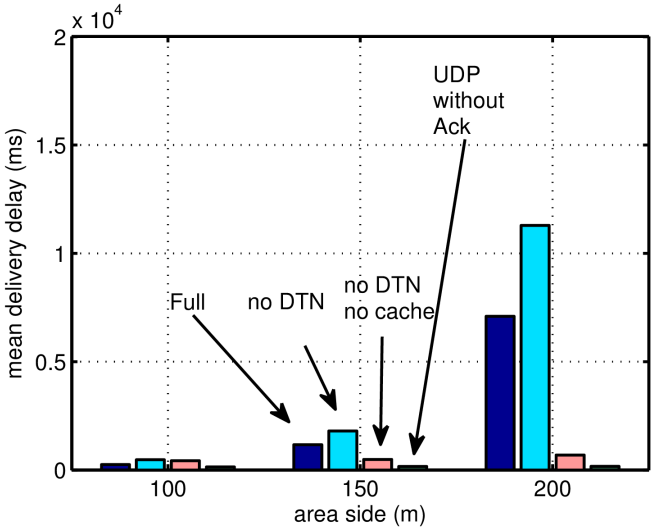

Fig. 19 – Mean delivery delay, unreliable mode, 2 topics, 15 nodes

VIII. CONCLUSIONS

The use of an Information Centric Networking technology like CCN simplifies the development of topic-based publish-subscribe systems, while providing valuable performance in terms of service reliability and latency in critical mobile environments, like MANETs. Indeed, data muling, caching and multicasting are very useful in these conditions and they are built-in in CCN.

Pull and Push are the two dissemination models on which publish-subscribe CCN systems may be based. The Pull approach can be developed by using the CCN architecture without modifying it, but requires polling, which in general leads to an undesirable overhead. However, in case of MANETs, polling is used also to refresh the topology of the dispatching multicast tree, which otherwise should be done with other means. Following this consideration, we have developed a pull based publish-subscribe system, namely TPS-CCN. The performance evaluation of our TPS-CCN system shows that the effectiveness of the CCN functionality increases with the area side and with the number of subscribers, i.e. with the distribution scale. And both these aspects are promising.

ACKNOWLEDGMENTS

This works is partially founded by BEE SAFE, a POR Puglia (Italy) research project lead by Sistemi Software Integrati, a Finmeccanica Company, by the EU-JP co-funded project GreenICN (FP7 grant agreement N. 608518 and NICT Contract N. 167) and by the EU CONFINE project (FP7 grant agreement N. 288535). We thank Claudio Pisa for the implementation of the CCNInfo OLSR plugin.

REFERENCES

[1] R. Buyya , M. Pathan and A. Vakali, "Content Delivery Networks", Springer, 2008

[2] D. Trossen, M. Sarela, K. Sollins, "Arguments for an Information-Centric Inter-networking Architecture", ACM Computer Communication Review, April 2010

[3] M. Gritter and D. R. Cheriton. TRIAD: A New Next-Generation Internet Architecture. Technical Report, July 2000, http://gregorio.stanford.edu/triad/

[4] V. Jacobson, D. K. Smetters, J. D. Thornton et al., "Networking named content", proc. of ACM CoNEXT 2009

[5] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, "Supporting the Web with an Information Centric Network that Routes by Name", Elsevier Computer Networks, vol. 56, Issue 17, p. 3705–3722, 2012

[6] M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: named-data link state routing protocol", proc. of ACM SIGCOMM 2013, ICN workshop, Hong-Kong, August 12, 2013

[7] CCNx project web site: www.ccnx.org

[8] Object Management Group, "Data Distribution Services for Real Time Systems", Version 1.2, OMG, January 2007

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, A. Kermarrec, "The Many Faces of Publish/Subscribe", ACM Computing Surveys, vol. 3, issue 2, 2003

[10] A. Carzaniga, M. Papalini, and A. L. Wolf, "Content-based publish/subscribe networking and information-centric networking", proc. of ACM SIGCOMM 2011, ICN workshop, Toronto , Canada, 19 August 2011.

[11] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, K.K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System," proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2011

[12] OpenPGM implementation https://openpgm.googlecode.com/files/libpgm-5.2.122.tar.bz2

[13] BEE DDS website http://www.beedds.com

[14] Common Open Research Emulator (CORE) website http://www.nrl.navy.mil/itd/ncs/products/core

[15] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt, "On the design of content-centric manets", proc. of WONS'2011 , Bardonecchia, Italy, Jan. 2011.

[16] R. Baldoni and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," DIS, Universita' di Roma "La Sapienza", Tech. Rep., 2005

[17] Huang, Yongqiang, and Hector Garcia-Molina. "Publish/subscribe tree construction in wireless ad-hoc networks.", Mobile Data Management, Springer Berlin Heidelberg, 2003

[18] Baehni, Sèbastien, Chirdeep Singh Chhabra, and Rachid Guerraoui, "Frugal event dissemination in a mobile environment" Middleware 2005, Springer Berlin Heidelberg, 2005

[19] Y. Huang, H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment", proc. of 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), 2001

[20] E. Anceaume, A.K. Datta, M. Gradinariu, G. Simon, "Publish/subscribe scheme for mobile networks", proc. of the 2002 Workshop on Principles of Mobile Computing (POMC 2002)

[21] Picco G.P., Cugola G., Murphy A.L., "Efficient content-based event dispatching in the presence of topological reconfiguration", proc. of 23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA

[22] G. Cugola, E. Di Nitto, A. Fuggetta, "The JEDI event-based infrastructure and its application to the development of the OPSS WFMS," Software Engineering, IEEE Transactions on , vol.27, no.9, pp.827,850, Sep 2001

[23] Source code of OLSR CCN routing tool, http://netgroup.uniroma2.it/Andrea_Detti/OlsrCCN/OLSR-CCN-Routing.tar.gz

[24] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout, " Java Message Service," Sun Microsystems Inc., Santa Clara, CA 2002.

[25] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks", proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05). ACM, New York, NY, USA, 252-259

[26] Kevin Fall, "A delay-tolerant network architecture for challenged internets", proc. of the ACM SIGCOMM 2003

[27] T. Clausen, P. Jacquet, "Optimized Link State Routing Protocol", IETF RFC 3626

[28] OLSRd project website, www.olsr.org

[29] Varvello, Matteo, et al. "On the design of content-centric MANETs", proc. of IEEE Wireless On-Demand Network Systems and Services (WONS), 2011

[30] A. Marica, A. Molinaro, and G. Ruggeri, "E-CHANET: Routing, forwarding and transport in Information-Centric multihop wireless networks", Elsevier Computer Communications, 36(7), 2013, pp. 792-803.

[31] J. Ott, D. Kutscher, and C. Dwertmann, "Integrating DTN and MANET routing", Proceedings of the 2006 SIGCOMM workshop on Challenged networks. ACM, 2006.

[32] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, L. Querzoni, “Structure-less content-based routing in mobile ad hoc networks”, proc. of IEEE International Conference on Pervasive Services (ICPS'05), 2005.

[33] P. Costa, C. Mascolo, M. Musolesi, G.P. Picco, “Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks”, IEEE Journal on Selected Areas in Communications, vol.26, no.5, pp.748,760, June 2008

[34] L. Wang, A. K. M. M. Hoque, C. Yi, A. Alyyan, B. Zhang, “OSPFN: An OSPF based routing protocol for Named Data Networking”, University of Memphis and University of Arizona, Tech. Rep. available at http://new.named-data.net/wp-content/uploads/TROSPFN.pdf, July 2012

[35] Ren, Zhong, Mohamed A. Hail, and H. Hellbruck. "CCN-WSN-A lightweight, flexible Content-Centric Networking protocol for wireless sensor networks", proc. of IEEE Eighth International Conference of Intelligent Sensors, Sensor Networks and Information Processing, 2013

[36] L. Junhai, Y. Danxia, X. Liu, F. Mingyu ”A survey of multicast routing protocols for mobile ad-hoc networks”, IEEE Communications Surveys & Tutorials, vol.11, no.1, pp.78-91, First Quarter 2009

[37] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou; C. Tsilopoulos, X. Vasilakos, K. Katsaros, G. Polyzos George, "A Survey of Information-Centric Networking Research," Communications Surveys & Tutorials, IEEE , vol.16, no.2, pp.1024-1049, Second Quarter 2014

IX. APPENDIX I: ADDITIONAL PERFORMANCE RESULTS

A. Comparison with IP reliable multicast

In this section we compare the TPS-CCN "Full" configuration operating in reliable mode with IP reliable multicast. IP reliable multicast is a traditional way to support topic-based publish-subscribe with IP means. Each topic is associated to a multicast group and to a related multicast distribution tree. Publishers send information items through IP multicast packets that are received by subscribers that belong to the multicast distribution tree. We implemented this IP multicast distribution by using the OLSRd BMF plugin and the OpenPGM tool [12], an open source implementation of the Pragmatic General Multicast (PGM) protocol. Both tools are used with their default configuration and we used the "daytime" command of OpenPGM to generate and send information items. OpenPGM is the only implementation of a distributed one-to-many delivery system that we could find available on the Web, even if other multicast protocols, better tailored for MANET, have been proposed (e.g. [36]), as well as topic-based publish-subscribe systems.

In all tests all published items have been indeed delivered. Fig. 20 shows that, with respect to IP multicast, TPS-CCN reduces the latency as the coverage area increases, i.e. when network connectivity decreases. This reduction is mainly due to CCN in-network caching and TPS-CCN data muling.

Fig. 21 reports the packets transmitted by the devices per delivered item. When the network is likely fully connected at one hop (i.e. in the 100m case), IP reliable multicast generates the lowest amount of packets, since it exploits MAC broadcast frames, while TPS-CCN uses UDP/IP unicast sockets (unless when in DTN mode) and MAC unicast frames. As the area increases, the network is more fragmented, the number of retransmissions increases and in-network caching makes the retransmission path of TPS-CCN shorter than the one of IP reliable multicast, which is necessarily rooted at the source. This reduction of path length tends to compensate the inefficiency of using unicast sockets and the difference between the number of transmitted packets decreases.

Fig. 22 and Fig. 23 report the same performance measurements versus the number of topics. Also in this case the latency of TPS-CC is better than the one of IP reliable multicast, while the number of transmitted packets per delivered item is similar.

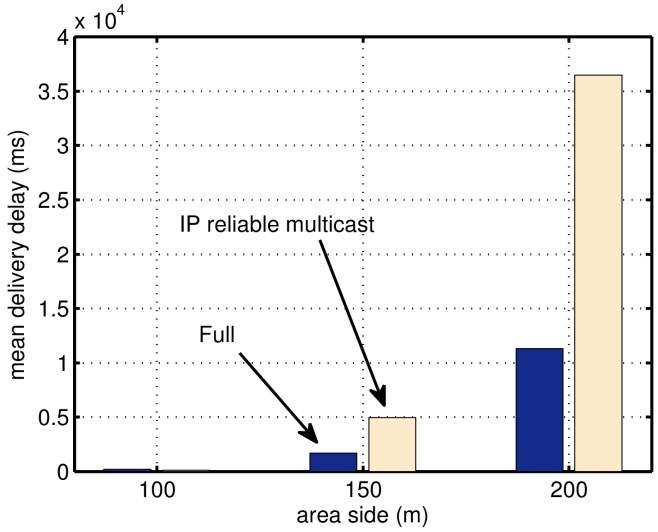

Fig. 20 – Mean delivery delay, reliable mode, 2 topics, 15 nodes

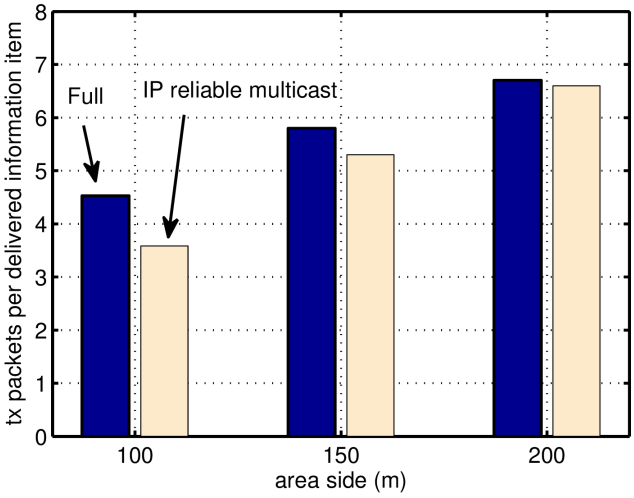

Fig. 21 – Number of transmitted packets per delivered information item, reliable mode, 2 topics, 15 nodes

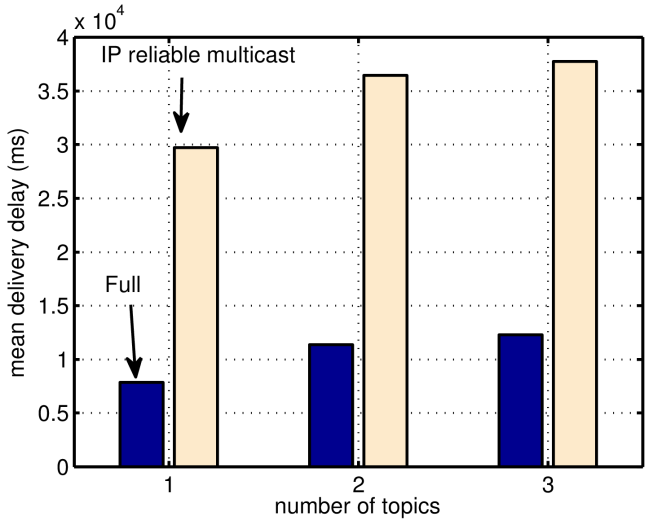

Fig. 22 - Mean delivery delay, reliable mode, 200m x 200m, 15 nodes

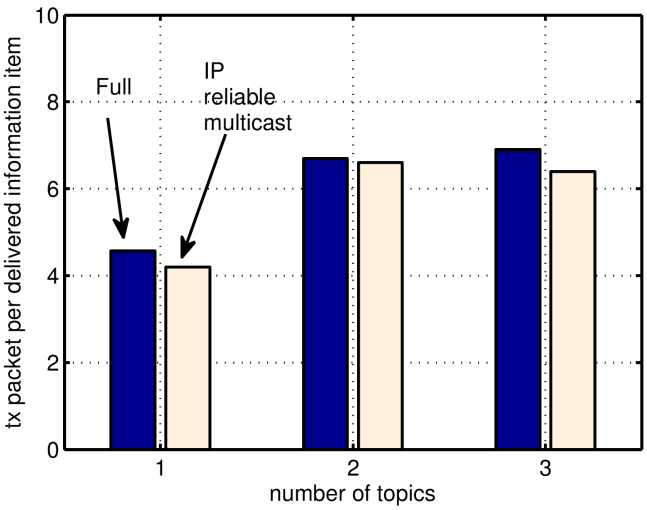

Fig. 23 – Number of transmitted packets per delivered information item, reliable mode, 200m x 200m, 15 nodes

B. Analysis of receiver window

In this section we analyze the impact of the receiver window size *W* used by subscriber flow control mechanism on the performance (Fig. 7). We report the simple case of a system with one topic and TPS-CCN "Full" configuration operating in unreliable mode. The derived observations also hold for system with two and three topics and in case of reliable operative mode, although related results are not reported here.

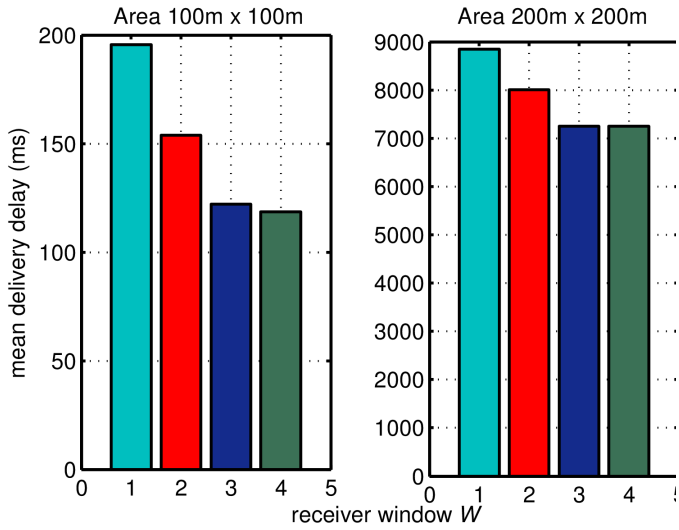

Fig. 24 - Mean delivery delay, unreliable mode, 1 topic, 15 nodes versus receiver window *W*

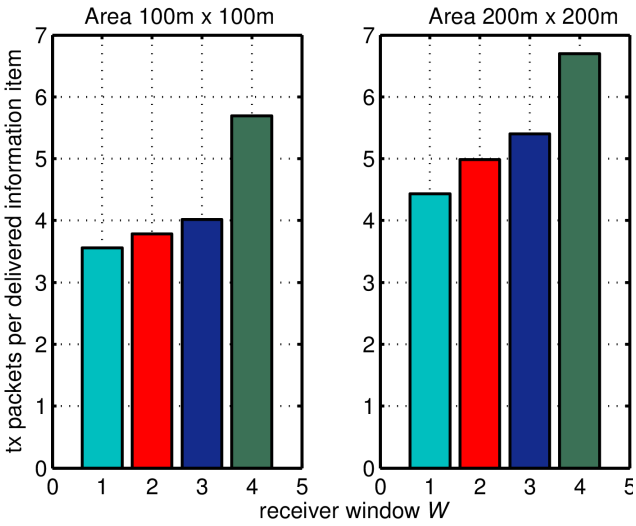

Fig. 25 - Number of transmitted packets per delivered information item, unreliable mode, 1 topic, 15 nodes versus receiver window *W*

Fig. 24 reports the mean delivery delay versus *W* with an area size equals to 100m and 200m. As the receiver window *W* increases, the delay tends to decreases, but beyond *W*=3 the delay reduction is very limited. Fig. 25 reports the number of transmitted packets per delivered information item. By increasing *W* we have an increase of this merit figure, i.e. a reduction of the delivery efficiency, since subscribers maintain in the network an higher number *W* of Interest, which are periodically resent to refresh the publisher-to-subscriber multicast dispatching tree. We note a quick growth between $W = 3$ and $W = 4$. Comparing Fig. 24 and Fig. 25 we observe that *W*=3 provides a delay/efficiency trade-off.

C. Additional results on delivery efficiency

In this section we report the number of transmitted packets per delivered information item measured in the cases examined in sections VII.A and VII.B. As discussed in section VII.A, these results confirm that TPS-CCN full configuration provides valuable results in terms of delivery efficiency.

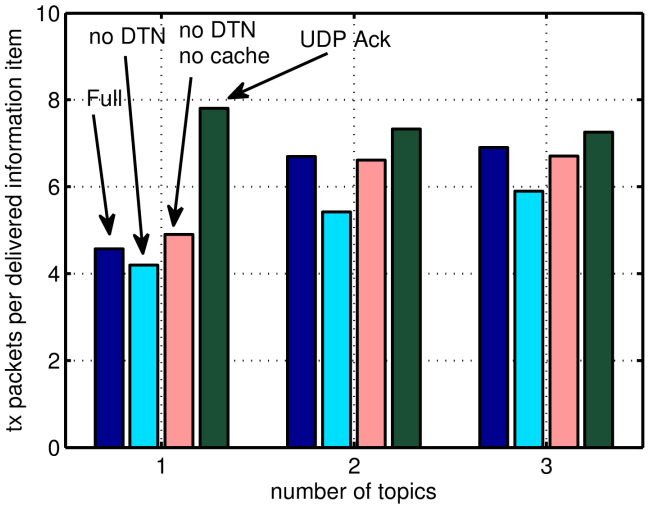

Fig. 26 – Number of transmitted packets per delivered information item, reliable mode, 200m x 200m, 15 nodes

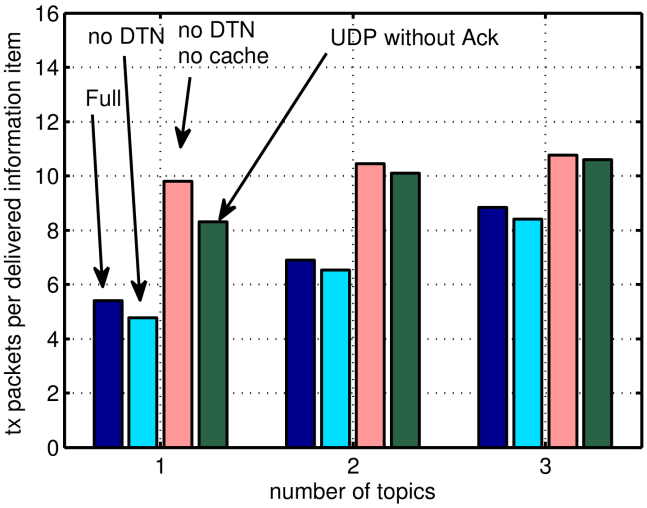

Fig. 27 – Number of transmitted packets per delivered information item, unreliable mode, 200m x 200m, 15 nodes

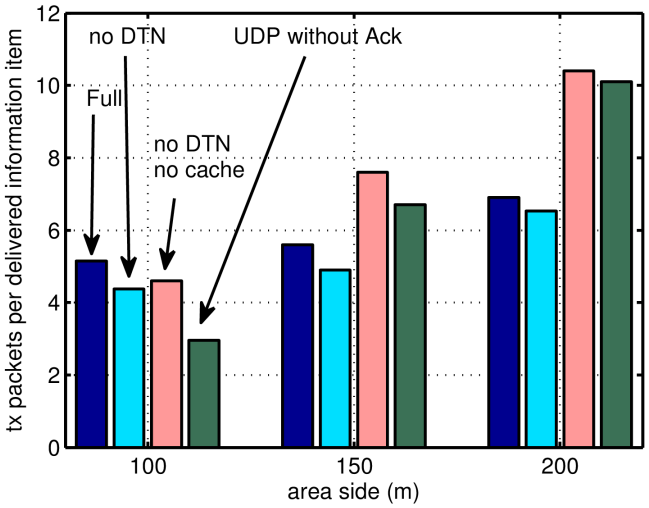

Fig. 28 – Number of transmitted packets per delivered information item, unreliable mode, 2 topics, 15 nodes