ABSTRACT
The information centric paradigm assumes that the Future Internet will be built around contents rather than host locations. In this direction, the promising Content Centric Networking (CCN) architecture has been conceived to solve the problems of the today Internet by means of a novel way for distributing contents (based on their names) and by adopting distributed caching mechanisms. CCN can be particularly beneficial in an ad hoc networking environments, where the main goal is the delivery of data to a given destination node no matter its position, and moreover, in scenarios characterized by limited connectivity, just as wireless ad hoc networks. Some interesting works, available in literature, propose the exploitation of CCN paradigm in Mobile Ad hoc Networks or Vehicular Ad hoc Networks. Our contribute, here, is to provide the scientific community with a new fully customizable and open source emulation platform, named CCN-Joker, well-suited for wireless devices with limited resources. The rationale of the developed tool entails an application-level overlay. It is composed by several modules interacting with the data structures which a CCN node is commonly made of. We also conduct some preliminary experiments in a small overlay (composed by six EFIKA powerPC boards, equipped by a Wifi external card), analyzing the content distribution process, the hit ratio, and the average download time. These preliminary results confirm the suitability of CCN-Joker to study the CCN performance in Mobile Ad hoc Networks or Vehicular Ad hoc Networks.

Keywords
CCN, Experimental testbed, wireless ad-hoc networks.

1. INTRODUCTION
Despite the uncontested success of Internet, the way the network is used is changed: information itself has become more and more important in all aspects of communication. In fact, most of the today Internet traffic is related to content distribution, which includes file sharing, media streaming, and so on. This information-centric use of the worldwide network raises new challenges, many of them not being handled effectively by the current network architecture [1].

For this reason, many research groups all over the world are discussing about the big topic of the Future Internet, which has to meet the requirements of scalability, heterogeneity, security, mobility, QoS, robustness, energy efficiency, economic incentives, and so on [13].

Several proposals have been formulated to face all these issues and most of them have as common factor a “data-centric” approach: what becomes more important then where. In other words, the contents can be exchanged independently of physical location of hosts and the storage for caching information becomes part of the network infrastructure.

The main data-centric architectures proposed for the Future Internet are the Publish Subscribe Internet Routing Paradigm [7], the 4WARD NetInf project [12], the Cache-and-Forward Network Architecture [4], the Data Oriented Network Architecture [6], and the Content Centric Networking (CCN) [5].

The last one, which is a very promising solution, lies on the basic idea that users can retrieve contents in a secure way without having any awareness about the physical location of the server, verifying their validity and integrity.

Several research groups are investigating about CCN exploitation in different contexts. In particular, as discussed in Sec. 3, CCN can be particularly beneficial in an wireless ad hoc environments to avoid routing loops and guarantee energy efficiency [10].

Our contribute, here, is to provide the scientific community with a new fully customizable and open source emulation platform, which will be referred to as “CCN-Java Opensource Kit EmulatoR” (CCN-Joker). It allows the emulation of the basic aspects of a CCN, such as the handling of Interest and Data packets, as well as cache sizing and replacement policies.

We have chosen to develop our CCN-Joker emulation platform instead of using the well known CCNx tool for several reasons: (i) we need a lightweight application program able to run on top of devices with limited resources (in terms of storage capability and computational load); (ii) we want to have the full control on cache management operations; (iii) we need to handle a customizable content requesting process driven by their popularities. Considering our purposes, the use of the CCNx platform would have been too much onerous, in terms of computational resources and nodes configuration. Moreover, we have preferred the Java language
due to the simplicity and the effectiveness of its networking library (i.e., java.net) and of its multi-threading programming support.

Moreover, we conduct some experiments on a real testbed using small PowerPC devices; in particular, we used Elika boards which are small, but complete and powerful computers, equipped with a wireless network interface card and a 2GB flash memory. The obtained preliminary results demonstrate how CCN-Joker allows the investigation of different performance indexes, such as the contents diffusion process, the hit ratio, and the average download time.

The rest of the paper is organized as follows: in Sec. 2 and 3 we summarize the main features of CCN and some interesting related works; Sec. 4 describes CCN-Joker emulation platform; in Sec. 5.1 we present the experimental testbed and in Sec. 5.2 preliminary experimental results are presented and discussed. At the end, Sec. 6 summarizes the main achievements of this contribution and draws future research directions.

2. CCN OVERVIEW

According to the CCN vision, communications are always receiver-driven and a name-based routing is adopted [5]. There are only two type of messages exchanged into the network: Interest and Data. A user interested in a particular content diffuses an Interest packet; each node having a copy of the required data replies with a Data packet. Other nodes just forward the Interest. In the Interest packet the only mandatory field is the ContentName, i.e., the name of the required item. A Data packet has an arbitrary length and it includes the name of the content, some additional information, and obviously the encapsulated payload.

CCN adopts a hierarchical structure for names, which leads to a tree-based resolution process. A CCN name is formed by several components: an Interest can specify the full name of the content or its prefix, thus accessing to the entire collection of elements under that prefix.

All CCN operations are made possible by three main structures: (i) the Content Store (CS), which is the cache memory of a node; (ii) the Forwarding Information Base (FIB), used to forward Interests towards potential sources of data, even through multiple interfaces of the same node at the same time; (iii) the Pending Interest Table (PIT) that keeps track of Interest packets already forwarded upstream in order to properly deliver Data packets downstream.

When an Interest packet arrives at a CCN node, the CS is searched to discover whether a data item is already available as an answer to be immediately sent back to the requesting user. Otherwise, the PIT is consulted to find out if others Interest packets, requiring the same content, have been already forwarded towards potential sources of the required data. In this case, the arrival interface of the Interest is added to the PIT entry. Otherwise, the FIB is examined to search a matching entry, indicating the list of interfaces the Interest has to be forwarded through. At the end, if there is not any FIB entry, the Interest is discarded. On the other hand, when a Data packet is received, the PIT table comes into play, and, keeping track of all previously forwarded Interest packets, it allows to establish a backward path to the node that has requested the data.

Another key aspect of CCN is the distributed caching. In fact, each node receiving a Data packet decides whether cache it, according to the adopted caching technique. In this way, the contents diffusion process is simplified, allowing a reduction both of servers computational load and bandwidth consumption [5].

3. RELATED WORK

A wireless ad hoc network is a collection of wireless nodes that can dynamically self-organize an arbitrary and temporary communication network without necessarily using any pre-existing infrastructure. In ad hoc networks, each node may communicate directly to each other. Nodes that are not directly connected communicate by forwarding their traffic through intermediate nodes. Every ad hoc node acts as a router. Ad hoc networks can be easily set up, even in desert places and can endure to natural catastrophes and war. These characteristics make ad hoc networks well suited for military activities, emergency operations, disaster recovery, large scale community networks, and small networks for interaction between meeting attendees or students in a lecture room. The design of an ad hoc network has to take into account several interesting and difficult problems due to noisy, limited-range, and not-secure wireless transmissions added to mobility and energy constraints [14].

All the features presented in the above section make CCN particularly beneficial in an ad-hoc networking environments, because, thanks to content-oriented networking operations, the overall energy efficiency of the system can be improved [8]. This is also confirmed by several recent works exploring the exploitation of CCN in Mobile Ad hoc Networks (MANETs). Some of them account for theoretical analysis of different design choices [15]. Other ones, instead, propose enhanced CCN implementations to face the peculiar requirements of MANETs [2][11]. For example, in [2] a novel architecture is conceived to enable broadcast packets, a local forwarding decision strategy, and new techniques to face mobility management, which are crucial in IEEE 802.11 based MANETs. Along the same line, the CCN paradigm is tailored to tactical and emergency MANETs in [11], introducing features and requirements for disruptive networks. Finally, in [3] a novel network architecture is designed by extending and modifying the CCN proposal in order to solve the main issues of hybrid Vehicular Adhoc Networks (VANETs).

Unfortunately, all these works lack of an experimental study in a real MANET, thus motivating the present contribution.

4. CCN-JOKER

The CCN-Joker rationale entails an application-level overlay that can be applied to both emulation-based analyses and real experiments. It is composed by several modules interacting with the data structures which a CCN node is commonly made of (see Fig. 1).

In addition to these well-known CCN structures (such as CS, FIB, and PIT [5]) CCN-Joker introduces two ancillary libraries:

- Repository: a permanent memory where the “seed” copies of the contents are stored;
- MyRequests: a dynamic table that stores the pending requests performed by the node itself.

The core module of CCN-Joker is an UDP server, namely JokerServer, listening on the port 9700. It is active on all
nodes of the overlay and its role is to handle incoming requests generated by other CCN-Joker modules, as shown in Fig. 2.

**Figure 2: JokerServer operations.**

In particular, a JokerServer instance can distinguish between internal and external requests. The former ones are generated by other CCN-Joker modules, working on the same node of the considered instance. They include the JokerClient, which provides an interface towards end-users, through which they may ask for contents, and the CacheManager, which is adopted to query the CS and discover the cached items. The latter ones, instead, embrace Interest and Data packets sent by other CCN nodes.

When an Interest is received, the JokerServer assigns its management to the RequestManager module, which behaves as illustrated in Fig. 3. At first, the RequestManager searches for a potential duplicate into the InterestTable. In the case of a positive outcome, the Interest is discarded. Otherwise, if the Interest lifetime is not expired, the required content is searched into both the CS and the Repository. If this research has a positive outcome, a Data packet is created and sent as reply. On the other hand, when a Data arrives, the JokerServer solicits the ContentManager. Its operation are presented in Fig. 4. In particular, if the content is already present in the CS or in the Repository, it means that the incoming Data is unsolicited, so that the ContentManager will ignore it. A matching PIT entry, instead, indicates that there are some pending Interest packets, thus the Data is sent to all the faces listed in the PIT entry and the content is added to the Cache. If the ContentStore is saturated, a Least Recently Used (LRU) replacement policy is adopted. Finally, the ContentManager checks into the MyRequest table whether the arrived content can satisfy a local request generated by the JokerClient. In case of negative response, the Data is discarded.

**Figure 3: RequestManager operations.**

On the other hand, when a Data arrives, the JokerServer solicits the ContentManager. Its operation are presented in Fig. 4. In particular, if the content is already present in the CS or in the Repository, it means that the incoming Data is unsolicited, so that the ContentManager will ignore it. A matching PIT entry, instead, indicates that there are some pending Interest packets, thus the Data is sent to all the faces listed in the PIT entry and the content is added to the Cache. If the ContentStore is saturated, a Least Recently Used (LRU) replacement policy is adopted. Finally, the ContentManager checks into the MyRequest table whether the arrived content can satisfy a local request generated by the JokerClient. In case of negative response, the Data is discarded.

**Figure 4: ContentManager operations.**

In conclusion, CCN-Joker allows users to emulate and customize the basic aspects of a CCN node, embracing all handling mechanisms for Interest and Data packets, as well
as cache sizing and cache replacement policy. Thanks to its modular structures, it could be easily upgraded in the future to support further CCN properties and enhancements.

5. EXPERIMENTAL RESULTS

One of the principal difficulties in the Future Internet research is the lack of real workloads and validating scenarios. Indeed, in this paper we start emulating a content-centric scenario in a relatively small overlay (showed in Fig. 5), planning, at the same time, to extend our work to wider Internet-like scenarios.

5.1 Testbed

Our testbed is a small wireless ad hoc network, composed by six Efika boards equipped with 4Gbyte Flash memory and with a WiFi module. Efika devices, showed in Fig. 6, are small, but complete and powerful computers, based on an ATX motherboard and on a RISC PowerPC processor. These tiny devices target mostly embedded applications. They also have a very low power consumption and are completely silent. The main Efika motherboard specifications [9] can be summarized as follows:

- ATX board (153x118x38 mm);
- Freescale MPC5200B PowerPC SoC up to 466 MHz;
- 128 MB DDR RAM @ 266 MHz;
- 44 pin IDE connector;
- 1 PCI (33/66 MHz PCI 2.2);
- 10/100 Mbit/s Ethernet (Realtek 8201 Phyceiver);
- 2 USB ports (1.1);
- 1 RS232 Serial port D-SUB9;
- Stereo audio out, microphone and line input S/PDIF (Sigmatel STAC 9766 AC97);
- Infrared port (IRDA) from 2400 bps to 4 Mbps;
- RTC clock (power management on/off).

On each Efika PC a Linux operating system (OpenSuse 11.1 distribution, kernel 2.6.32) has been installed. Furthermore, each device has been equipped with an IBM developer kit for Java technology on PowerPC hardware and with a Linux Optimized Link State Routing Protocol. When CCN-Joker daemon starts, each Efika behaves as a CCN node. Due to the limited storage capability of these devices, CCN-Joker has been optimized to face this issue and to interact with both network interfaces (wireless and wired).

5.2 Preliminary Experiments

All the experiments have been conducted in a 50 m² area. However, to emulate a wider area nodes have been “deceived” about the their neighbors lists, in order to narrow the visibility range of each node.

There are six Efikas, in partial visibility each other, each one has a cache memory with limited size and it generates requests according to a Poisson process with parameter $\lambda$. Further, contents popularities follow a Zipf distribution with parameter $\alpha = 0.65$ (see Fig. 7).

The parameters adopted in our experiments are summarized in Table 1. As starting point of our study about the exploitation of CCN architecture in a wireless ad-hoc network, we investigate three metrics, listed below:

- average number of content copies: we average out the number of nodes having a cached copy of a specific content during the time (without considering the transient);
- hit ratio: calculated as the ratio between the hit events and the total number of requests;
- average download time: the average time between when a request starts from a given node and when the required data is obtained.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>6</td>
</tr>
<tr>
<td>Cache size</td>
<td>125 [contents]</td>
</tr>
<tr>
<td>Catalog size</td>
<td>2500 [contents]</td>
</tr>
<tr>
<td>Content size</td>
<td>10 Kb</td>
</tr>
<tr>
<td>Cache to Catalog ratio</td>
<td>0.05</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>1 [request/s]</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.65</td>
</tr>
<tr>
<td>Experiment duration</td>
<td>5 hours</td>
</tr>
</tbody>
</table>

Table 1: Main testbed parameters

In Fig. 8 the number of contents’ copies in the network is showed compared with a proportional trend. It is possible to note that the experimental trend tracks almost the proportional trend, obtained by the following expression:

\[ n_i = B \cdot S \cdot \frac{\pi_i}{\sum_{i=1}^{M} \pi_i} \]  

(1)

where \( B \) is the cache size, \( S \) is the number of nodes, \( M \) is the catalog size and \( n_i \) represents the number of copies for the \( i \)-th content.

Figure 7: Zipf distribution (a) and CDF (b).

Moreover, to better understand CCN dynamics, we observe the number of content copies for some contents during the time. By way of example, in Fig. 9 the copies diffusion for the most popular contents is reported. It is evident that the equilibrium point is approximately reached after 3 hours.

Figure 8: Content copies at equilibrium point vs. contents IDs.

At the end, we evaluate the the average download time for each content. The download time is defined as the time between the instant when a request for a particular content departs from a node and the instant when the required data arrives to the requesting node. Observing Fig. 11, it is possible to note, for a large number of contents, that the average download time is less then 2 s, but for other items it become larger, up to 12 s.

These results, which clearly demonstrate the utility of CCN-Joker in wireless ad hoc networks, cannot be compared to other ones due to the lack of experimental works about CCN MANETs. In [2] the download time is evaluated through a simulation-based study, starting from different hypotheses (TCP connection, content size 10 Mbyte), while in [11] the end-to-end delay between source and destination nodes is observed in a linear topology. This results will be enriched in the near future considering different metrics and a more realistic use case, such as disaster recovery.

6. CONCLUSIONS

With this work we present CCN-Joker, a new lightweight fully customizable and open-source emulation platform useful to emulate the basic aspects of a CCN node, such the handling of Interest and Data packets, as well as cache sizing and replacement policies. CCN-Joker has been designed...
to run on top of wireless devices with limited resources, such as Efikas or PowerPCs, in order to test the CCN exploitation in a wireless ad-hoc network. We also conduct some preliminary experimental tests in a small overlay, analyzing different metrics, such as the content distribution process, the hit ratio and the average download time. These preliminary results, that will be enriched in the near future and also compared with respect to the prior art, just confirm the suitability of CCN-Joker to study the CCN performance in MANETs or VANETs.

7. ACKNOWLEDGMENTS

This work was supported by the projects ERMES PON 01_03113/3, founded by the Italian MIUR, and "A Smart Cart for Customerized Shopping Experiences", Kad3 Laboratory, Italian Government, art. 1-law 106/2011.

8. REFERENCES


