ICON - An Information Centric Architecture for Opportunistic Networks

Bruno Batista
SITILabs, University Lusófona
Email: bruno.batista@ulusofona.pt

Paulo Mendes
SITILabs, University Lusófona
Email: paulo.mendes@ulusofona.pt

Abstract—In recent years large attention is being devoted to the development of information centric architectures. The major motivation has been the understanding of the mismatch between the current Internet architecture - host based - and the way users tackle the network - share and access information. The first proposals have been focus on fix infrastructures in an attempt to devise a global information centric Internet. However, since the number of available wireless Internet access points and the number of wireless mobile devices is growing significantly, it is also very important to have in mind the requirements of scenarios where networking is done by exploiting any communication opportunity when devising an information centric architecture. The motivation of our work is to devise an information centric framework compatible with the major architectural trends and able to allow operation in dynamic scenarios with wireless devices owned by anyone. Since everyone should be able to use it, the proposed framework aims to be versatile, multiplatform and easily to maintain.

I. INTRODUCTION

Nowadays most of us use the Internet for some kind of activity (business or pleasure) on a daily basis. The Internet moved from the bulky desktop computer to more pervasive devices like smart phones or some form of embedded device that, for instance, can be found in intelligent buildings, at home, and on the street (e.g. for monitoring purposes and advertising). The increasing capabilities of pervasive devices in terms of CPU, storage and communication means, on one hand that they can consume and produce large amounts of information, and on the other hand, exacerbate, problems related with mobility, with heterogeneity in terms of software (mainly operating system) and hardware, as well as operation in opportunistic networks, in which users take advantage of the diverse communication, computation, sensing, storage and other resources that surround them to produce and share information.

The exponential growth of the Internet exacerbates the fact that the way we use the Internet and network systems in general and the way those systems work are clearly different. People only care about obtaining information, while being agnostic of the location of such information. From the network point of view the location is essential to find the requested content, thus when someone asks for a content by name, that name has to be resolved to a location and a name that the network can identify. The current Internet architecture, very dependent upon content and hosts location, is far from ideal in scenarios where hosts are mobile, have a multitude of communication interfaces, which need to work in a seamless way, and may have intermittent connectivity, such as when embodied in an ad-hoc and opportunistic network (e.g. a delay-tolerant network). To address these, and other problems related with the dependency of content location, several Information Centric Network (ICN) architectures arose, among them Content Centric Networking (CCN)[1], Publish-Subscribe Internet Routing Paradigm (PSIRP)[2], Network of Information (NetInf)[3] and Data Oriented Network Architecture (DONA)[4]. These kinds of architectures represent a major shift in how networks work. Despite their differences, regarding conceptual and implementation approaches like naming convention, name resolution and security, there are similarities in key aspects like the focus on data in all aspects of the networking process (from routing to security) and the fact of considering the possibility that every node on the network has some memory space to cache information that passes through them. Another thing that they have in common is the fact that the first considered scenario was a wired access to the Internet. In the case of NetInf mobile access networks were considered, which introduced more instability due to the mobility of source and consumer of data. Nevertheless NetInf still distinguishes between static data forwarders and mobile data sources and consumers.

This paper is looking at more dynamic scenarios, where nodes that can be sources, consumers and forwarders of data and all of them are mobile and intermittently available. In this dynamic context, some effort has been done towards showing that ICN architectures can also handle real-time data[5], and work in more dynamic networks like MANETs[6]. [7] [8]. Although MANETs operate over scenarios more dynamic than the wired Internet, they assume that in any moment in time it is possible to find a path among any pair of nodes, which, in opportunistic networks, such as delay-tolerant networks, does not happen. In opportunistic networks pervasive devices exploit any contact opportunity to try to forward a given message considering some criteria (e.g. always forward, analyze the likelihood of finding the destination node, only forward if the node is the destination of the information) in a deterministic or probabilistic way. As far as we know there is no published research that tackles this precise scenario.

Summing up we are considering a use case with high diversity of networked devices that communicate in a network where the connectivity is intermittent, and where nodes can be
mobile, and can have multiple communication interfaces with very different technologies. It is a complex use case, but a real one, since nowadays personal devices are more and more used to setup communications in urban scenarios and in more challenge scenarios (e.g. critical-mission networks).

Hence, we believe that the next generation of information centric architectures should have some key characteristics: be able to run in very different devices; support different type of communication interfaces and protocols; be very flexible, in order to take advantage of any communication opportunity; be easy to maintain and upgrade. As far as we can tell CCN is the most suitable architecture for the described scenarios, because it does not need any centralized entity (for instance to do name resolution) in order to communicate, considers the possibility of a node to communicate through different interfaces and supports some customization of the node behavior by means of the strategy layer. So conceptually CCN meets some, but not all, of the criteria to operate over dynamic networks: it has a layered configuration that is simple and clean. However, the layered architecture is not very flexible, the strategy layer is limited in scope, and the real world implementation, CCNx, lacks the desired flexibility, upgrade-ability and maintainability.

Therefore, we propose a new multiplatform information-centric framework, called ICON, which has several conceptual similarities with CCN, allowing full interoperability, but is adapted to operate in dynamic wireless networks. Moreover, ICON follows a new software design allowing it to be easy to install, deploy and upgrade. The proposed architecture, as described in this paper, is available at SITILabs Technology web site (http://siti.ulusofona.pt/index.php/technology) to be used by everyone that would like to develop information-centric applications for pervasive wireless networks.

The remainder of this paper is organized as follows: In section 2 we describe the use case considered to evaluate ICON. Section 3 describes the ICON architecture and finally in section 4 and 5 we conclude and present our proposals for future work.

II. APPLICABILITY STUDY

In order to evaluate the real potential of ICON we consider a real world scenario were the mentioned requirements of flexible design and support for dynamic networking environments make a real difference over the existing information-centric architectures. Figure 1 illustrates an heterogeneous scenario that should be considered to evaluate the applicability of an information centric solution. This scenario is composed of four different networking environments: a fix global infrastructure, provided today by the Internet (upper right corner); a local structured wireless network used mainly for wireless access to the Internet (upper left corner); a network of embedded devices that can communicate directly among them or via a gateway (bottom left corner); a network of personal wireless devices that can communicate directly among them or via a gateway (bottom right corner). The first two networking environments (Internet and Local wireless LAN) are today fully operational based on an host-to-host IP network able to sustain access to data as well as real-time communications. The latter two networking examples (Internet-of-things and personal pervasive networks) are not fully deployed in the real world, mainly due to the limitations posed by the host-to-host communication model to pervasive operations over more dynamic scenarios.

If we look at the work done to develop a fully operational information-centric networking architecture, considerable effort is being dedicated to the Internet and wireless local access scenarios. However, by the natural use given to embedded and personal wireless devices, is access and sharing of data, the Internet-of-things and the network of pervasive personal devices scenarios are the ones, in our perspective, that have more to gain with the development of a flexible, easy to use information-centric framework able to cope with operations in dynamic scenarios. This provides the strategic motivation for the development of ICON.

If we consider a real world use case, we can envision the case of Peter, who works in a building with wired, wireless, and sensor networks. Peter normally uses his working station for any type of interpersonal communication (videoconference, voip calls) over the wired and wireless networks. For any data oriented activity, he likes to use his ICON enable personal wireless device, which allows him to be agnostic of any source of information (Google, local sensor, neighbor personal device). At lunch time Peter likes to read the daily news on is personal device, so he uses ICON to gather the latest sport news: normally the required information are collected via the local WiFi access point, but the cafeteria is in an area of the building with poor WiFi access, so ICON is able to opportunistically gather the data of interest to Peter from the nearest device, which in this case is Bill’s ICON enable smartphone. After lunch he comes back to his office and when he arrives he feels that the office is too hot, so he grabs his personal device to check, in the internal network, the temperature and state of the air conditioner, which is unsuccessful because of a failure in the internal network. This is no major problem since the sensors in his building communicate to each other via ICON, making use of bluetooth interfaces, so Peter is able to gather information directly from the nearest sensor device, which on its term had inferred the temperature and humidity status of the local area by sharing

Figure 1. Heterogeneous applicability scenario
data with other ICON enabled sensors. At the end of the day, Peter likes to go to the coffee shop to meet some friends. Since some of his friends share his interest in sport news, their personal devices immediately exchange the data that match these interest after each wireless contact opportunity. At home, Peter’s device collects information about the home environment by announcing a set of interests. After collecting the relevant data form the network, a local home automation application installed in his personal device controls the home environment by sending commands to home actuators (previously configured to receive data related to home commands).

The pervasive nature of the described scenario requires an information-centric architecture that is easy to install and maintain by any user, and that is easy to upgrade with new functionality, in order to sustain the desirable fast deployment pace of a more pervasive Internet. This motivates the creation of ICON based on a flexible software design approach, as described in the next section.

III. THE ICON ARCHITECTURE

The development of ICON started with the challenge of creating a framework that could be adapted virtually to any networking scenario, but that would include, from the start, opportunistic networks, which have not been tackled by prior-art. Opportunistic networks bring extra challenges since on the one hand protocol wise, we are talking about taking advantage of any communication opportunity through any possible interface, and on the other hand, the success of opportunistic networks depends on how easy it is for users to be part of it.

So, our initial requirement is to create an architecture able to grow easily and to be updated and maintained by everyone. First, it is clear that we need a modular software design in which every module is self contained and loose coupled in relation with the overall architecture. In this case we incorporate in ICON a modular approach, as the one used in Haggle[9], which works well in opportunistic environment; second, the architecture needs to be interoperable with other information-centric architectures, namely the ones derived from the focus placed by researchers on the Internet and wireless local networks. In this aspect, and due to its global impact, we decided to incorporate some of the CCN functions, allowing an ICON device, operating in a dynamic environment, to interact with a CCN node in a wired network.

A. ICON Concept

As can be seen in Figure 2 ICON is composed by three modules that we call Engines: Decision Engine, Data Engine and Network Engine. Inside of each one are other modules related with the specific tasks of each engine. This modular approach allows to properly compartmentalize each functionality of each engine and thus makes it easier to change, update and maintain.

The data engine is responsible for storing information related with the data that passes through the node and with the persistent data system. The network engine is responsible for managing all the communication interfaces and protocols: this is the module that actually sends the data. The decision engine it is the “brain” of the architecture, since it is the module that takes all the decisions, such as how the different modules relate to each other, or about the external behavior, for instance how the node relates with its neighbors. In Figure 2 there is also another module that represents how the node communicates with its neighbors. We call this module Protocol and it is composed by the Name, Interest and Content messages. The Name is used to identify data objects accordingly with the defined naming convention, which in this case is the same of CCN (is an URI with and hierarchical structure). The Interest and Content define respectively the message to express interest in a certain data object and the returned message with the data object itself, also inline with the CCN messages. The usage of CCN similar definitions of the protocol elements is done with the intention of ensuring interoperability.

Besides the operational engines, and aiming to allow an easy maintenance, every ICON devices come with a simple embedded web server to show the status of the node, namely the communication interfaces, the FIB and PIT tables, and the content store. This information is shown in real-time so it is possible to monitor the node operations.

1) Network Engine: Looking closer to the different modules we can see that the network engine is composed by two modules. The communication interface manager which is responsible to initialize and register the different communication interfaces (Ethernet, wifi, bluetooth, etc) that are defined on a XML file and that we will go over it later on this section. The communication manager is where is implemented the access to the interface hardware. We need to have one implementation for each type of interface and communication protocol.

2) Data Engine: The data engine is composed by four modules all of them based on concepts derived from CCN although the modular implementation used in ICON gives them more flexibility making it also easier to analyze and improve performance. The modules of the data engine deal with a lot of information and they have to do so without compromising the performance, but since their functionalities are adequately divided it is easy to identify problems and correct them. The content store manager is responsible for managing
how and where the node stores the data that passes through it. Currently the content store keeps the information in memory, due to performance issues related to search and deliver of the requested information. When a new interest arrives to a node, the first place (unless there is a rule that says otherwise) to look at is on content store. The default replacement policy is Least Recently Used (LRU). The content segments manager is responsible for segmenting the data to be send. The segment size is configurable and should be inline with the communication protocol to avoid double segmentation. In what concerns the serialization of the segments, ICON uses proto-buffers, because is cross platform and probably is the fastest and more efficient serializer available today (is a binary serialization). Proto-buffer was originally created by Google and it is used in most of the their communications. The FIB manager keeps track of the data forwarded by each active communication interface and the PIF manager keeps track of the interests that were not satisfied also grouped by interface.

3) Decision Engine: While the network and data engines are fully implemented, the first version of the decision engine is still under development. We are taking especial care with the development of the decision engine, because is the ultimate tool to adapt the behavior of the architecture to the operation of a dynamic network, and is where end users can implement their business logic. The decision engine encompasses four modules: rule management, human behavior inference, data synchronization, and context awareness.

The rule management module allows the introduction, at any moment, of rules that can alter the internal or external behavior of the node. These rules can be static, that is, the user establish a set of rules that can only be changed by him/her, or can be dynamic, that is, rules can be changed by the decision engine itself based of what it learn along some period of time, such as based on the output of the human behavior inference module. This to say that the rule management module is being developed based on machine learning algorithms that can grant some “intelligence” and consequently some degree of autonomy to the node (that is of particular importance in opportunistic networks). The rules should be introduced in some kind of human readable meta language, but that should not represent a big overhead to parse into the architecture. A preliminary implementation is already available, but still requires some testing to be validated.

The human behavior inference module provides an ICON device with social awareness and learning properties, making use of reality mining techniques integrated with network protocols and services. This functionality allows devices to use locally sensed social context information (e.g. via Wi-Fi, Bluetooth, email conversations) in order to infer behavior patterns (e.g. daily periods and location) of repeated communication (e.g. text messages, emails, instant messages) among recurring peers and infer affinity groups and relationships between local behavior and the global network structure. All these information is used to adapt the way ICON handles data exchange, such as by establishing rules to make data dissemination aware of social ties. Inferring processes must be build based on robust context classifiers, to mitigate the limitations of existing techniques in coping with inconsistent labels, which are inevitable in real-world scenarios, as the one described in this paper. Hence, ICON includes the development of active learning techniques to address a large-scale deployment.

The synchronization module has as its principal function the analysis of the behavior of neighbors nodes in order to determine if it is possible to collaborate by redistributing data in order to reduce network traffic and increase data delivery.

The context collector module acts as a support to all the other modules since its task is to collect information about the node performance and about everything that surround the node. This information is then used by the other modules to support decision making. Currently this module is implemented as an external middleware that allows ICON devices with different operating systems to store, make available and share in the same manner sensor data as well as the coupling of virtualized sensors that physically exist in other devices. The sensed data can be not only environmental (positioning, motion), but also social (wireless contacts), and local (battery, storage).

B. Software Design Choices

ICON development was done based on a set of major design requirements including the capacity to be cross-platform, flexible and easy to maintain and upgrade. Naturally these requirements influenced the choices that we made in what concerns the software development platform and the architecture itself. In order to be cross platform we need a programming language/platform that, with minimum effort runs in mobile devices, desktops and embedded devices. We also wanted to do something with good performance and not only a proof of concept. So we narrowed our choice to tree options: C++, Java and C#. C++ give us the good performance, but needs to be compiled to every platform that we decided to use. Moreover the development is slow since it is a “hard core” language. Java is a more productive language, but has some performance problems in some platforms, and the support in mobile devices is not straight forward in all platforms and very limited in embedded devices. So we decided to develop the software in C#. This allows us to seamlessly run the code in Windows, Mac and Linux machines thanks to the Mono framework (http://www.mono-project.com), to run the code with small changes in Windows Mobile phones with Microsoft .NET Mobile framework, Android phones with MonoDroid (http://xamarin.com/monoforandroid), iPhones with MonoTouch (http://xamarin.com/monotouch) and with a specific version on embedded devices running the Microsoft .NET Micro framework. This special version is needed because of the lack of support for generic types of the .NET micro framework. C# also has the advantage of being a modern up to date high performance language, free and standard (ISO/IEC 23270:2006), providing relatively fast developing and being widely used by industry and software companies. Also Mono and Microsoft .NET framework provide support for the most common communication interfaces and protocols. For those
that they do not support it is also possible to create them through the .NET/Mono, or with direct access to system and then wrapped to .NET/Mono.

ICON should also be flexible and easy to maintain and upgrade. To achieve this we structured the architecture in modules with very clear interfaces and use dependency injection to load them. The modules to load are specified by XML files and are loaded in run time rather than in compile time. From the alternatives that we had to do the dependency injection, we chose TinyIoC (https://github.com/grumpydev/TinyIoC) because of its simplicity and due to the fact that it works in every platform except in .NET micro framework.

It is also worth mention that ICON is a multi-threading architecture, which means that it can process several messages (interest or data) at the same time and without blocking each other.

All the choices that we take related with the software development and the architecture design were tested and validated in a test bed comprising 3 laptop with tree different operating systems (Windows, Linux and Mac OS). We defined a network topology in which node A is connected to nodes B and C, B is connected to C and A, while node C is connected with B and A. This way we can verify how simple it is to configure a node and the routes (FIB), test the load and unload of modules, verify the flow of messages (interest and content), verify the data delivered and verify the overall performance of the architecture. Regarding the performance, it is difficult to ascertain how fast it is because we do not have a benchmark to compared with, although we verify that the architecture is multi-thread.

1) Node Configuration: Like already mentioned the basic configuration is done using XML files, such as the node configuration (c.f. Figure 3) and the interface configuration files (c.f. Figure 4). As can be seen in Figure 3 the name of the DLL’s of the engines is a parameter. So in order to change each one of the engines it is only a matter to pointing to the new DLL (of course that the defined interfaces for each engine should be respected). The others parameters indicate which is the name of the file that contains the interfaces definitions (Figure 3), the default directory to put the requested content, the maximum size of a data segment, the size of the socket buffer (by default the node communicate thru a UDP socket), the minimum and maximum time before resending an interest message, the time to resend interests and the interest timeout, and lastly the address of the web server to check the node state.

In the XML file related with the communication interfaces we can define the address of the interface and the port that it uses to communicate. We repeat the process to all the interfaces of the node and for the interfaces of its neighbors to where the node can forward data. Next we define a set of rules to control the interfaces, namely: what are the interfaces that the node uses to receive data (it can be one or more); what to do with specific prefixes. For instance, the last block (Forward) is where we fill the initial FIB, so what we are saying is that all the interest with the prefix “ccn4d:Images” received in the interface with the key 1 should be forwarded to the interface with key 2.

Of course that in a real implementation the FIB and the interfaces of the neighbors would be set dynamically based on the outcome of a routing protocol, and not static like in this example. It should be noted that the architecture is fully prepared to have the interfaces and FIB dynamic; the XML is just a convenient way to define routes and interfaces in the prototype (is a proof of concept), and does not influence the behavior of ICON. Currently we are developing an information-centric version of the dLife opportunistic routing protocol[10], [11], which will allow ICON nodes to automatically set forwarding rules based on the social environment of the nodes.

IV. ICON USABILITY

Based on the requirements set for ICON development (e.g. easy to use, flexible), we need to evaluate how easy it is to develop an application on top of ICON. ICON can be instantiated with one or any combination of three different roles: router, producer of information and consumer of information. To setup a information-centric router, we just have to import the ICON DLL and initiate the node as illustrated in the code snippet on Figure 5. To setup a producer of information we need to initiate also the ICON node, indicating which rule to use to fetch content: look only at the ICON content store or try to get the content from another local source. In the case of the snippet on Figure 5 ICON also looks at a specific location in the hard drive, but the source could be other (for instance real time steam). Finally, to setup a consumer of information, the
 ICON node needs to be initiated by indicating which method to use when requested data is received: by default data goes to the defined directory in the hard-drive, but ICON can also feed it directly to an application, which is useful to devise real-time applications.

The snippet illustrated on Figure 5 shows how easy it is to use ICON to setup consumers, producers and forwarders of information. The three role can coexist in the same device.

V. Conclusion

In this paper we propose an information-centric architecture for dynamic networking scenarios, allowing the interoperability with CCN (selected as a success case of an architecture running over a wired network) over wired networks. Our major motivation was the lack of an ICN framework for pervasive personal devices. Such ICN architecture needs to fulfill not only the networking requirements related with the exchange of information over dynamic networks (e.g. data synchronization, opportunistic routing, human behavior inference), but also with software design requirements to sustain the fast pace deployment of ICN in pervasive scenarios. With this goal in mind, we presented the ICON architecture, which we believe have all the important properties to work on dynamic networks of pervasive devices, which should be able to exploit any wireless contact opportunity to allow information to be exchanged, to inter-operate with CCN and to grow alongside with the appearance of new communication technology due to its design flexibility, easy maintenance and upgrade. ICON flexibility is shown for instance in the support given to different communication interfaces, by simply implementing a different interface in the communication manager and updating a configuration file. Such updates can be done in runtime allowing an easy adaptation of ICON to changes in the node behavior and context. Actually, by taking advantage of the proposed declarative rule management module in the decision engine, we can make our node “intelligent” enough to take its own decisions. ICON is easy to maintain and upgrade because everything is in well define and self-contained modules, making it very user friendly for fast application development. Finally, ICON multi-threading capability, ensures a high performance standard.

VI. Future work

The current ICON implementation is ready to be used by everyone that wants to make use of the basic features of ICN. In the future we will proceed with the investigation and development of the functionalities that we consider the most important to allow an ICN framework to operate in a dynamic network, namely the three modules present in the proposed decision engine for rule management, data synchronization and inference of human behavior.

There is also much to be done regarding the evaluation of ICON. Currently we are deploying a testbed with real nodes in a real situation, as the one we describe in the use case. In order to make the evaluation scenario scalable, we combine real equipment, with mobility and traffic models simulated in MatLab, and ICON nodes running on virtual machines. This way we can test the new concepts and also the implementation in terms of performance and robustness.

References


