Codecast: Code-based routing and delivery in Delay-Tolerant Networking

Álvaro García Cantalejo

Abstract– In this paper, a new code-based, dynamic adressing scheme called Codecast has been implemented over the aDTN platform of the SeNDA research group. This adressing scheme allows the inclusion of C source code on DTN bundles which is later executed to determine the recipients of the bundle. This will allow to improve communications in challenged networks on inaccessible or remote areas, or danger zones. The result was an implementation on the said platform which allows for dynamic code-based bundle routing and delivery. This will enable further developments in this area to create other addressing schemes based on Codecast's flexibility.
Keywords– Codecast, active-DTN, DTN, addressing scheme, delivery, mobile-code.

Resumen– En este trabajo, un nuevo esquema de direccionamiento dinámico basado en código llamado Codecast se ha implementado sobre la plataforma aDTN del grupo de investigación SeNDA. Este esquema de direccionamiento permite la inclusión de código fuente C en bundles DTN que más tarde se ejecuta para determinar los destinatarios del paquete. Esto permitirá mejorar las comunicaciones en redes desafiadas en zonas inaccesibles o remotas, o zonas de peligro. El resultado fue una aplicación sobre dicha plataforma que permite el enrutamiento y entrega de bundles basado en código dinámico. Esto permitirá nuevos desarrollos en este área para crear otros esquemas de direccionamiento basado en la flexibilidad de Codecast.
Palabras clave– Codecast, Active-DTN, DTN, esquema de direccionamiento, entrega, código móvil.

1 Introduction

1.1 Context

THERE exist certain situations where traditional networks lose effectivity. For instance, when considering rural settings with no pre-existing infrastructure, it might be inviable to establish a traditional network, due to the inability to connect to the Internet or establish costly infrastructures.

However, in this kind of settings, another type of networks might be set up. DTN (Disruption or Delay Tolerant Networks) [2] are opportunistic networks characterizd by intermitent conectivity, asymmetric bandwidth, high and variable latency and arbitrary mobility patterns. These kind of networks are based on the assumption that there isn't a stable connection between all members of the network. In fact, it's possible to establish a DTN among nodes that aren't connected between them, given the assumption that network members will eventually come into contact with each other.

In this kind of disconnected networks, nodes communicate between them with bundles, which are literally bundles of messages which are grouped together to be more easily sent. This grouping is managed by a new protocol called Bundle Protocol [7] which operates in the Application Layer. This protocol allows transmission in DTN networks since the rest of layers are oriented to constant transmission which cannot be adapted to the intermitent service of a DTN network. Thus, the sender can send multiple messages in a single bundle; allowing the Bundle Protocol to see inferior layers as a transmission method in DTN networks.

In a DTN network adressing and routing are performed differently than in traditional networks. Given the intermitent connection between endpoints, physical directions are pointless, so directions in a DTN network are merely arbitrary identifiers. This constraint makes it impossible to route messages in the traditional way, which is one of the most important problems in DTN environments. Thus, the only possible strategy to route a message through a DTN network is to disseminate it throughout the network and the efficiency of this process will depend on the dissemination strategy.

1.2 Motivation

As we can see, DTN networks are interesting because they allow networks to exist where they previously could not,

E-mail de contacte: alvaro.gcantalejo@gmail.com
Menció realitzada: Tecnologies de la Informació
Treball tutoritzat per: Sergi Robles (Departament d'Enginyeria de l'Informació i les Comunicacions)

as we can see in [1]. It is in this kind of challenged networks that we can find an opportunistic take on network communication, allowing messages to be relayed without a pre-existing infrastructure. This allows to bring networks even to challenged areas or danger zones to provide better communication where there is currently none.

However, we can find two main problems in DTN networking. In the one hand, we find the routing problem, or how to decide which route should bundles take to reach their destination. On the other hand, we find the delivery problem, or how to decide to whom the bundle should be delivered when we can't identify it unequivocally.

Thus, to help improve DTN networks and allow them to improve, we find the need for a new mechanism which can solve these problems to improve DTN communications. This project will focus on developing a new code-based, dynamic adressing scheme named Codecast to solve the delivery problem in opportunistic networks.

1.3 Objectives

For all this, this project will try to implement a new addressing scheme called *Codecast* starting with the base of the aDTN platform of the SeNDA investigation group of the dEIC [11].

Project objective Develop a code-based routing and delivery mechanism over the aDTN platform of the SeNDA investigation group.

To accomplish this objective, the project will be divided in three main objectives, shown below:

Objective 1: Analyze the viablity of the implementation of Codecast over the aDTN platform.

The analysis has as objective to determine the requirements to implement Codecast, as well as checking which ones are already in the platform.

Finally, we will be able to determine if it is possible to realize this implementation, its difficulty and which tasks are necessary to carry them out.

Objective 2: Design and implement Codecast over the aDTN platform.

Based on the requirement list determined in the analysis, it will be necessary to create a model of the behaviour of Codecast that allows to extrapolate a design, and finally implement the design over the aDTN platform. Likewise, it will be necessary to realize performance tests to verify the correctness of the implementation.

Objective 3: Realize tests and simulations to test the behaviour of Codecast over systems with big amounts of endpoints.

Once implemented Codecast, a series of proof of concept tests will be created to be executed in a DTN simulation to get results on Codecast's behaviour.

All this will allow us to determine if Codecast is a reliable and efficient mechanism.

2 State of the Art

The most significant contribution to the study of DTN stems from the IRTF investigation group on Delay-Tolerant Networks. This group defined on 2007 the underlying architecture of a DTN in the RFC 4838 [2], and the Bundle Protocol in the RFC 5050 [7]. These RFC introduce the concept of bundle as a set of data blocks that allow applications to progress, as well as the *store-carry-and-forward* paradigm, in which every endpoint can store data to relay to other nodes when contact is established.

Clare, Burleigh y Scott, establish in [3] how the bundles of a DTN are routed through arbitrary identifiers that are not required to have topologic nature. Thus, a DTN can use any addressing scheme it wishes, since conventions have not been established on which schemes should be used and how they would interact between them.

There also have been some studies on the use of mobile code in DTN environments. In [1], a new paradigm called *store-carry-process-and-forward* is proposed, in which mobile code is used to implement a DTN in a grid to provide access to a network of sensors intermittently connected. Thus it is achieved a system that dynamically adapts to intermittent disconnections and solves the problem of routing. A similar proposal is found in [5] using a message relay, but this proposal does not propose an architecture or specific implementation.

Finally, we also find several proposals on group communications in DTN environments. In [6], the concept of group-based communications in DTN using the Manycast paradigm is proposed. In [4], a study of Multicast in DTN is presented, comparing it with the Unicast paradigm and formulates the relay selection problem for Multicast as a knapsack problem.

3 Methodology

For this project a waterfall methodology with regression and prototyping has been employed, providing methods to return to an earlier stage if identifying a problem. This methodology breaks down the project into tasks which are performed in a linear order, one at a time. After each task, a backup of the project's state is made, so that changes can be reverted if they need be.

The project has been broken down into three main tasks, which are the feasibility study, the design and the implementation. The feasibility study includes the requirements analysis and the platform review, and intends to establish the bases of what needs to be done to carry out the project. The design includes all of Codecast's behaviour, on how it operates over the platform and how it carries out it's functions. The implementation comprises the actual coding of the design into the platform.

For the feasibility study, all of Codecast's possible features have been identified, and have been subdivided into individual features as specific as possible. These features make up the requirements, which have been divided into functional requirements and nonfunctional requirements, according to whether they attend a function or specific criteria for assessing the system.

For the design, the Universal Modeling Language (UML) has been used to elaborate a flow chart, a sequence diagram and use case diagram, because it provides methods to model the behavior unequivocally applications. The flow chart represents Codecast's program flow, and how it operates normally. The sequence diagram accurately depicts the interaction between modules. The use case diagram describes each use case and the modules that participate in it.

To ease the implementation, it has been broken down in different subtasks as performed due to the design, it is relatively easy to implement on existing code on the platform. The subtasks that the implementation has been divided into consisted of the modification of a single module individually, so the chances of causing an error in the existing code are reduced, and if there was one it would be easier to find because of this. Once the modification of a module is completed, a white-box testing ensues to ensure the correct behaviour of the module and its outputs are as expected.

After finishing the entire implementation, a phase of module integration has followed, in which each module has been adapted to ensure that the behaviour as a whole is the expected; and a white-box testing has also been made on the platform together with several bundles to verify that the bundles are routed and delivered properly according to the destination code and routing code.

4 DEVELOPMENT

4.1 Feasibility Study

The feasibility study consists of two main parts. On the one hand, the requirement analysis, in which the requirements to implement Codecast are listed; and on the other hand, the review of the aDTN platform, in which the requirements which are already found on the aDTN platform are listed, and therefore need not be implemented.

The requirements will be broken down into functional requirements, which specify what Codecast must be able to do and what features it should have; and nonfunctional requirements, which indicate how Codecast must fulfill the functional requirements, but do not express a feature it should have.

The requirement analysis yielded the following requirements for the implementation of Codecast:

Functional Requirements:

- Ability to send bundles through the network.
- Ability to embed C source code in bundles sent over the network.
- Ability to embed different code fragments in the same bundle. These fragments must be of the following types:
  - Lifetime code
  - Routing code.
  - Destination code.
- Shared data region that allows communication between code fragments.
- Ability to receive bundles sent over the network.
- Ability to store bundles received in a data structure.
- Ability to individually process bundles stored following a preset policy.
- Ability to run the code embedded in messages.
- A routing list indicating which neighbors the message should be sent to must be built.
- A recipient list indicating the applications the message should be delivered to must be built.
- Ability to define whether the message will continue stored in the data structure or dropped.

Nonfunctional requirements:

- Source code must be written in C.
- Embedded code in messages must be written in C.
- The code must be processed quickly and efficiently.
- It should be a robust system.

We can see that there is a vast majority of functional requirements, with a few non-functional requirements. This is because Codecast's functionality must be fragmented into as many requirements as possible to ease the design and implementation phases, whereas those requirements that are not from their functionality are rather scarce.

Having established the requirements that are necessary to implement Codecast, it has undertaken a review of the aDTN platform. After analyzing the code of the platform, it has been found that it already meets the following requirements:

- Ability to send bundles through the network.
- Ability to embed C source code in bundles sent over the network.
- Ability to embed different code fragments in the same bundle. These fragments must be of the following types:
  - Lifetime code
  - Routing code.
- Ability to receive bundles sent over the network.
- Ability to store bundles received in a data structure.
- Ability to individually process bundles stored following a preset policy.
- Ability to run the code embedded in messages.
- Source code must be written in C.
- Embedded code in messages must be written in C.
- It should be a robust system.

As shown, the aDTN platform already fulfills a large majority of the requirements for implementing Codecast, so the amount of functionality to be implemented is largely reduced. Therefore, it can be said that the implementation of Codecast on the aDTN platform is feasible based on the requirement analysis performed.

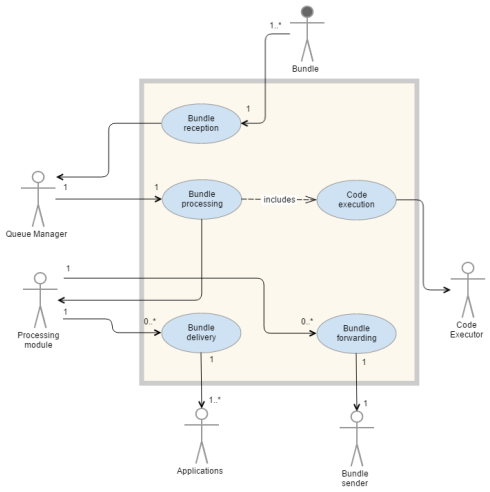

Figure 1: Use Case Diagram

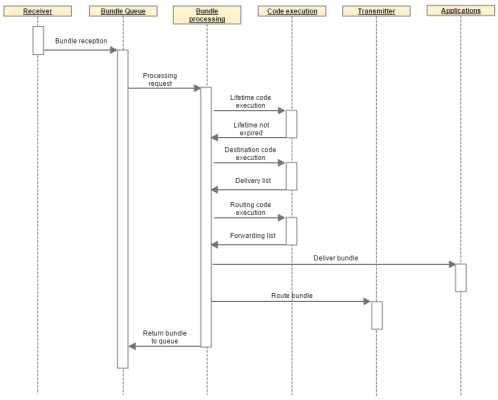

Figure 2: Sequence Diagram

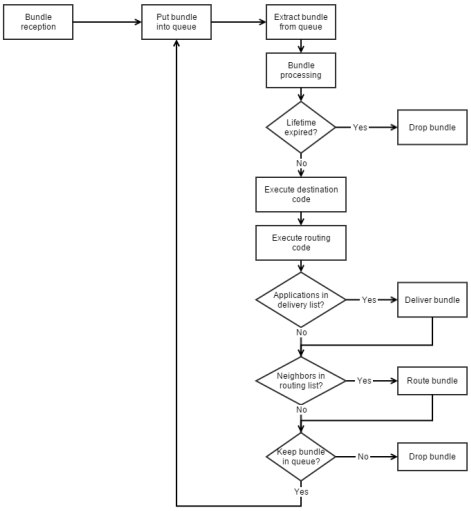

Figure 3: Flowchart

4.2 Design

In this section, the design of Codecast will be explored. This design has been made to reflect its global behaviour. Codecast's behaviour has been considered for several use cases (see Figure 1). It is a complex process consisting of multiple use cases, which are described below.

The first use case is bundle reception. bundle reception runs asynchronously in a Receiver module. Each Receiver will listen for bundles from nearby nodes and each time it receives one, it will be stored in a bundle queue. This reception is run separately from the rest of the application, so that it can still be run even if multiple simultaneous bundles are received or Codecast is processing a bundle at the time.

This queue will be managed by a Queue Manager module, which is responsible for ordering bundles within the same. This queue can store these bundles for an indefinite period. The Queue Manager will extract the bundles from the queue orderly following the queue management policy defined for the system.

The next use case is bundle processment, which is performed after removing a bundle from the queue. It consists of several steps (see Figure 2). First, the bundle is removed from the queue and sent to the Processing module, which is responsible for processing bundles and for communicating with the Executor module to send bundle code execution requests. This use case also includes the code execution use case, which is performed by the Executor module under demand of the Processing module.

The code execution use case is performed by the Executor module. This code is always executed in the same order to maintain consistency in the performance of the system (see Figure 3). Firstly, the lifetime code is run, and determines whether the bundle's life has expired. This code can directly remove the bundle from the Processing module if the lifetime has expired.

Then, if the bundle has not expired, the data in the shared data area, which is a DTN metadata block [9] in which destination and routing codes can write and read data to be stored and shared among them . Thus, they can communicate with each other, even between different network nodes. These data are passed as parameters to both the routing code and the destination code.

Subsequently the destination and routing codes are run. First is run the destination code, which receives as a parameter a list of applications which are subscribed to the platform. From this execution, a delivery list is built, which is a list of applications the bundle should be delivered to, according to each application's parameters. Then is run the routing code, which receives as a parameter a list of available neighbors. From this execution, a routing list is built, which is a list of neighbours the bundle should be forwarded to.

Both the routing code and the destination code can affect the permanence of the bundlein the queue, allowing them to return a boolean variable which indicates if the bundle should remain in the queue or not. In this case, due to the order of execution, the routing code's decision overwrites the destination code's decision.

Finally, once the code has been run, the system proceeds to carry out the decisions, which comprises the last two use

cases. In the delivery use case, the Processing module delivers the bundle to those applications in the delivery list. In the forwarding use case, the Processing module sends the bundle to the Sender module to forward it to those neighbors in the routing list. Finally, the bundle is dropped if it is decided that it shouldn't continue in the queue. Otherwise, the bundle is put back in the queue waiting to be processed again. This allows bundles to be persistent in the system and can be delivered or forwarded multiple times.

4.3 Implementation

In the implementation, Codecast's design has been carried out over the aDTN platform. Because the platform had already implemented much of the functionality required by Codecast, it was decided to maintain the existing source code and make additions or modifications thereto.

First, the Bundle module has been modified to add the constants that define the type of destination code and data area to the existing code types. This enables the addition of destination code and shared data zone to bundles generated by the platform. Subsequently, a default destination code has been created temporarily, that simply delivers the bundle to the first application on the list. This allows destination code to be run even if it is not included in the bundle.

After the inclusion of the destination code as a new type of code, it has been necessary to modify the Executor module so that destination code can be run, as it behaves differently depending on the type of code to execute. For this, the Worker module and the Executor executor have been modified to load all the parameters needed to run the destination code, as the list of applications, and aDTNDHelper module has been created, which is a dynamic library, and is linked with the destination code to provide the list of applications to be returned.

Then the logical flow of the program has been modified to include destination code execution. First, the Receiver module has been modified; previously it performed the delivery function by checking the ID of the recipient directly with the identifier of the platform before placing the message in the queue. As decided in the design, delivery functionality lies with the Processor module, so it was removed from the Receiver module.

Subsequently the processing of the bundles in the Processor module has been changed so that when processing a bundle first modifies the RIT, a tree-like structure that stores information about the local aDTN platform, loading temporarily in the branch /data/¡bundleID¿ the shared data area so that it can be modified at runtime. Then it runs the destination code, and then the routing code, and from these executions, receives the delivery list and the routing list. Once it has finished running the code, the bundle is delivered to those applications in the delivery list and sent to those neighbors in the routing list.

Finally the API of the platform in the aDTN module has been changed so that when a DTN socket is bound the branch /application/¡appname¿ is also inserted into the RIT, which describes the application, and serves for the application to insert its own data that wants to offer to the destination code execution.

5 Results

Once concluded the implementation of Codecast on the aDTN platform, it has been checked for for proper performance, using different tests. First, the code has been compiled using the platform's CMake, properly edited to include new modules and how they are linked, and have been found to compile correctly on 64-bit Unix-based operating systems.

Then, there have been white-box testing in the aDTN platform to verify the implementation of Codecast working properly and modules are integrated among themselves as they should. These tests have been successful, and it has been observed that the modules work properly together.

Finally, there have been several black-box testing to test several bundles bundles that are routed and delivered properly according to the destination code and routing code. These tests have been conducted on the default codes defined in the platform, and have been successful, as the codes have behaved as expected.

The results obtained indicate a correct performance of Codecast within challenged networks. Previously, all delivery in opportunistic networks has been performed as strictly Unicast, since the Bundle Protocol [7] standard defines the delivery condition by comparing the destination identifier in the bundle with the platform identifier.

However, with Codecast, new kind of addressing schemes may be used, since it allows for any kind of code snippet to be inserted, which allows to address any kind of delivery condition. This might allow us to replicate, for instance, Manycast behaviour in a DTN network, as proposed in [4]. This would allow for more efficient routing and delivery in opportunistic networks. Therefore, the results obtained are considered satisfactory with respect to the initial objectives of the project, as Codecast works correctly in the aDTN platform.

6 Discussion

The results show that the implementation of Codecast works correctly with simple conditions. Codecast allows the use of dynamic routing schemes that are determined by code, as opposed to traditional addressing schemes in which the set of receivers is statically determined.

However, other considerations should be taken into account. In this section Codecast restrictions and possible features that could help improve it will be explored. First, the limitations of Codecast should be noted. Codecast is limited firstly by the size of code to be included in the bundle. The larger the code size, the greater the time it takes to transmit, build and run.

Another restriction is that Codecast runs any code included in the bundle. This could make it vulnerable to attacks including malicious code in the bundle. Since entering the code analysis field to determine malicious intent would be very difficult, it would be interesting to have a scheme similar to PGP in which codes that are trusted to be executed could be signed, or even other endpoints for total trust on all the codes they send. This would add a layer of security to Codecast that it now lacks.

Another limitation of Codecast is the way how it may interact with other addressing schemes, such as unicast or

multicast. Since Codecast determines recipients based on code execution destination, it is only compatible with other addressing schemes that it can operate over, for example, Multicast, so that only the endpoints subscribed to a particular multicast address actually run the Codecast code to see if they are recipients of the Bundle.

Finally, we should also consider the practical uses of Codecast. Theoretically, different addressing schemes allow better dissemination of messages through the network. However, let us take a practical case to set an example. Message routing and delivery in in rural areas with no existing infrastructure is very difficult, because there isn't an underlying network that allows the rapid transmission of messages, and these rural areas tend to be large and sparse. An emergency in such areas could go unnoticed without any means of communication. However, with Codecast it would be possible to disseminate messages quickly through opportunistic routing and delivery in these areas employing a code which delivers through geolocation, as proposed in [10], enabling quick communication in areas where previously it wasn't possible.

7 Conclusions

The objective of this project was to develop Codecast, a code-based mechanism for bundle delivery on challenged networks, over the aDTN platform of the SeNDA research group. We consider that this objective has been met, since the Codecast has been implemented on the aDTN platform and works satisfactorily.

The requirements specified during the requirements analysis have also been completed for the project, and has been obtained as a result a functional software that can route and deliver bundles by the code embedded in the bundle itself, as it was intended.

As a result, the student has obtained a much more detailed knowledge in software design and requirements analysis; as well as C programming and, specifically, in the environment of the aDTN platform, which has given him broad knowledge about the workings of memory and code building in low-level languages, as well as communication in asynchronous networks.

It is worth mentioning that the project has had to deviate from the plan due to time constraints and complications in the time schedule, which have prevented the test and simulation phase from being carried out to verify the performance of Codecast over other methods of routing in challenged networks.

7.1 Future lines

It should also be noted that the project still has the following to develop future lines:

- Tests and simulations to verify the behaviour of Codecast.
- Implementation of different routing and destination codes to create different distribution patterns.
- Implementing a code-signing PKI over Codecast to avoid malicious code injection.
- Codecast implementation over the new version of the aDTN platform in C ++.
- Improved API offered to the destination code through the dynamic library.

Acknowledgements

Firstly, I would like to thank Sergi Robles for his patience and encouragement with this project.

I would also like to thank my friend Marc Palenzuela for giving me ideas and lending a hand when needed.

And last, but not least, I would like to thank my mother and my father for all their support when I most needed it.

References

[1] Carlos Borrego and Sergi Robles. *A store-carry-process-and-forward paradigm for intelligent sensor grids.* Information Sciences, 222(0):113 – 125, 2013.

[2] V Cerf, S Burleigh, A Hooke, L Torgerson, R Durst, K Scott, K Fall, and H Weiss. *RFC 4838, Delay-tolerant networking architecture.* RFC 4838 (Informational), 2007.

[3] Loren Clare, Scott Burleigh, and Keith Scott. *Endpoint naming for space delay/disruption tolerant networking.* In *Aerospace Conference*, 2010 IEEE, pages 1–10. IEEE, 2010.

[4] Wei Gao, Qinghua Li, Bo Zhao, and Guohong Cao. *Multicasting in delay tolerant networks: a social network perspective.* In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 299–308. ACM, 2009.

[5] Qun Li and Daniela Rus. *Communication in disconnected ad hoc networks using message relay. Journal of Parallel and Distributed Computing*, 63(1):75–86, 2003.

[6] Samuel C Nelson, Yih-Chun Hu, and Robin Kravets. *Anycast, multicast and beyond: The role of manycast in dtn communication.* 2011.

[7] K. Scott and S. Burleigh. *RFC 5050, Bundle protocol specification.* RFC 5050 (Experimental), November 2007.

[8] David L Tennenhouse, Jonathan M Smith, W David Sincoskie, David J Wetherall, and Gary J Minden. *A survey of active network research. Communications Magazine*, IEEE, 35(1):80–86, 1997.

[9] S Symington. *RFC 6258, Delay-Tolerant Networking Metadata Extension Block.* RFC 6258 (Informational), 2011.

[10] Adrián Sánchez-Carmona, Sergi Robles, Carlos Borrego, and Gerard Garcia-Vandellós. *PrivHab: A Multiagent Secure Georouting Protocol for Distributing Podcasts in Disconnected Areas*. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[11] Carlos Borrego, Sergi Robles, Angela Fabregues, Adrián Sánchez-Carmona. *A mobile code bundle extension for application-defined routing in delay and disruption tolerant networking*. In *Computer Networks*, Volume 87, 20 July 2015, Pages 59-77.