Softwarecast: a code-based Delivery Manycast scheme in Heterogeneous and Opportunistic Ad Hoc Networks

Carlos Borrego[a,*], Gerard Garcia[a], Sergi Robles[a]

[a]*Departament d'Enginyeria de la Informació i de les Comunicacions*
Universitat Autònoma de Barcelona
Barcelona, Spain

Abstract

In the context of Opportunistic Ad Hoc Networking paradigms, group communication schemes (Manycast) are difficult to conduct. In this article, we propose a general delivery scheme for Manycast group communications based on mobile code. Our proposal extends network addressing by moving from the static header field paradigm to a software code-based addressing scheme. We allow messages to be delivered using built-in software codes that consider application-defined, context-aware or history-based information. Additionally, we allow messages to carry a delivery state that permits them to perform refined delivery-decision-making methods. As a consequence of this scheme, we have found that new group communication schemes, besides the traditional ones, may be beneficial to improve the network performance and to provide new functionalities to emerging scenarios like intermittently connected networks of heterogeneous physical objects. We present an application of this scheme to solve, following an analytical delivery method, the problem of sending a message to k and only k nodes of a heterogeneous and opportunistic network scenario that fit best a given criterion. We show, using simulations, that our proposal performs better than traditional approaches. Finally, to show that our proposal is feasible, we present an implementation of our proposal in a real Opportunistic Ad Hoc network, a DTN network, compatible with the de facto standard Bundle Protocol.

Keywords: OppNet, Ad Hoc, Bundle Protocol, Mobile-code, Manycast, Secretary Problem

1. Introduction

As portable devices, such as mobile phones, tablets, vehicles, physical objects, and other items are being widely deployed, new network paradigms are emerging. In contrast to traditional managed networks, these devices may be connected using Ad Hoc Networks that lack central infrastructure. In Opportunistic Ad Hoc Networking (OppNet) [12], these network contacts are intermittent and nodes are spontaneously pair-wise connected.

OppNet messages may be intended for a group of application instances in different nodes from the network. The group communication scheme that allows communication with an arbitrary number of group members specified by the sender is called Manycast. It includes different schemes such as Anycast, Multicast and Geocast [8]. In OppNet, these group communication schemes are still open fields of study. Traditional approaches such as the ones employed in connected networks are difficult to apply: group membership distribution trees are difficult to maintain during the lifetime of group sessions.

In this article, we propose a general scheme for Manycast group communication schemes, and we apply it to OppNet. As a consequence of this scheme, new group communication schemes, besides the above-mentioned, may be achieved. This scheme improves OppNet performance and provides new functionalities to emerging scenarios like intermittently connected networks of heterogeneous physical objects, a special case of Internet of Things (IoT) [39].

Our proposal extends network addressing by moving from the static header field paradigm to a software code-based addressing scheme. We allow the very same messages to carry software codes to make the delivery decision. That is, the decision on which applications should receive the application information. These decisions can be made in terms of application-defined, context-aware or history-based information. This is especially advantageous when the network has a large number of heterogeneous nodes and the applications need dynamic and elaborate ways of selecting message destinations.

We show, as an example scenario to prove the potential of our proposal, a network solution to the max-k-node delivery problem, that is, the problem of sending a message to k elements in a network that fit best a given criterion. The solution to this problem is tackled from a statistical perspective, very similar to other problems such as the *Secretary problem* [17, 15], a well-known statistics problem in the context of search theory.

Additionally, a wide set of simulations using a simulator capable of reproducing our proposal is presented. The results of these simulations show that our proposal performs better than the state of the art ones since it allows new group communication behaviours that are capable of being addressed using optimal strategies.

Finally, we present an implementation of this novel paradigm, that is compatible with the Bundle Protocol [31], and allows applications to send bundles to refined software-code-based Manycast destinations beyond the traditional IP-like Multicast groups. This implementation considers different security policies in order to detect malicious software codes.

The main contributions of this paper are: (a) Softwarecast: a general routing and delivery scheme based on mobile code for OppNet. (b) A statistical solution to the max-k-node delivery problem for $k = 2$ for OppNet using Softwarecast. (c) A secure implementation of this scheme in the context of DTN, and compatible with the *de facto* standard Bundle Protocol.

The rest of the paper is organized as follows: Section 2 reviews the state of the art. In Section 3, Softwarecast, a general group scheme for Manycast delivery is presented. Section 4 describes a practical application of our proposal. Section 5 covers the experimentation of our proposal using simulations. In Section 6, an implementation of our proposal in the context of the Bundle Protocol is described. Finally, Section 7 presents the diverse conclusions that are drawn.

2. State of the art

Opportunistic Networks (OppNet) are networks characterised by intermittent network contacts, asymmetric bandwidths, long and variable latencies and ambiguous mobility patterns. Messages are opportunistically routed via intermediate nodes from the source to the destination.

A very interesting case of OppNet is Delay and Disruption Tolerant Networking (DTN), a paradigm defined by the IRTF Delay Tolerant Network Research group. They introduced the DTN architecture (RFC 4838 [9]) and the Bundle Protocol (RFC 5050 [31]) that are an abstract service description for the exchange of messages in intermittently connected networks. In DTN, messages are called Bundles and are seen as a series of contiguous data blocks containing enough semantic information to allow the application to make progress where an individual block may not.

These Bundles carry the application information from a source to a destination following the store-carry-and-forward paradigm. That is, each node stores application data that can forward whenever the node contacts another node. The Bundle architecture behaves as an overlay network.

Concerning DTN addressing, in the Bundle Protocol, as described by Loren et al. [10], the endpoint identifiers are the tokens by which DTN Bundles are routed to their destinations. These tokens are not necessarily addresses because neither the scheme name nor the scheme-specific-part is required to have topological significance. An Endpoint Identifier's (EID) scheme-specific-part may have topological significance, depending on the definition of the named scheme, but alternatively, it may be a name or an expression which must be evaluated in some way in order to be converted into a name or address (or multiple names or addresses). A DTN system can use any URI scheme it chooses, and there are as yet no real conventions as to how different schemes could be used and how they might interact.

Several proposals for solving group communications in DTN networks have been already presented. Wu et al. [40] classify DTN Multicast into three categories. On one hand, in the single node model, a single node is responsible for the Multicast delivery. This model normally has high transmission delays. On the other hand, in multiple-copy and single-copy models, several nodes deliver Multicast messages. These models aim to control the communication overhead and buffer occupancy. Nelson et al. [25] approach the concept of group-based communication in DTNs by exploring the delivery scheme of Manycast is presented. Moreover, Gao et al. [18] present a study of Multicast in DTNs with single and multiple data items, they investigate the essential difference between Multicast and Unicast in DTNs, and formulate relay selections for Multicast as a unified knapsack problem by exploiting node centrality and social community structures. Yang et al. [41] propose a node-density based adaptive Multicast routing scheme that can handle different network scenarios than the existing Multicast delivery schemes for DTNs. Asplund et al. [1] present a partition-tolerant Manycast algorithm for disaster area networks that uses a random walk gossip protocol. Zheng et al. [42] propose SemanticCast, a content-based data distribution approach over self-organizing semantic overlay networks. This proposal maintains a self-organizing semantic overlay based on view exchange. Finally, Soares et al. [32] propose a routing protocol which takes routing decisions based on geographical location data, and combines a hybrid approach between multiple-copy and single-copy schemes.

However, the above-listed group communication strategies are limited to traditional network delivery schemes such as Multicast, Broadcast, Unicast, Geocast, etc. When facing more complex delivery strategies like *send-to-the-best-n-nodes* or *send-to-all-nodes-that-satisfy-x*, these proposals fail to give good general purpose and dynamic network-layer solutions. As a consequence of this limitation, some applications are not possible to be deployed in OppNet.

Mobile code is a good technology to implement these strategies. Borrego et al. [4], for example, present an infrastructure in the context of grid computing based on mobile code to describe grid resources not only taking into account the resources themselves, but also other resources of the same type. Using primitives like "*best*" they improve several grid services such as the information service and the monitoring service.

More recently, there are interesting studies using mobile code in OppNet scenarios. Borrego et al. [5] propose a new paradigm called *store-carry-process-and-forward* that uses mobile code to improve the integration of wireless sensor networks and grid computing infrastructures. Additionally, Borrego et al. [3] present a general purpose, multi-application robot-sensor network based on mobile agents that run mobile code on Mobile-C platforms [26]. This intelligent system was deployed in a concrete DTN scenario. However, the resources needed by the mobile agents makes it inappropriate for many other scenarios. Moreover, Borrego et al. [6] present a solution to improve DTN performance that consists in extending the messages being communicated by incorporating software code for forwarding, lifetime control, and prioritisation purposes.

Furthermore, several proposals in DTN have been presented to include a message state information in DTN messages to improve DTN issues such as message routing. For example, Spyropoulos et al. [33] present the popular Spray and Wait routing protocol that *sprays* a number of copies into the network, and then *waits* till one of these nodes meets the destination. The number of times a message can be replicated is included in the message and it is decreased every time the message is *sprayed*. This is a very good example of message state. Unfortunately, these message states are not able to be implemented in the Bundle Protocol: there is no current way of including such state in the Bundle Protocol messages.

Even though there has been considerable work on OppNet Manycast, there are many complex group delivery transmissions that are still very difficult to conduct and may be very useful in many OppNet scenarios. In the next sections, we define a novel OppNet Manycast scheme to solve this problem.

3. Softwarecast: a general group scheme for Manycast delivery

In the context of OppNet, applications use the intermittently disconnected network to exchange information. Nodes may host instances of these applications that create, or may

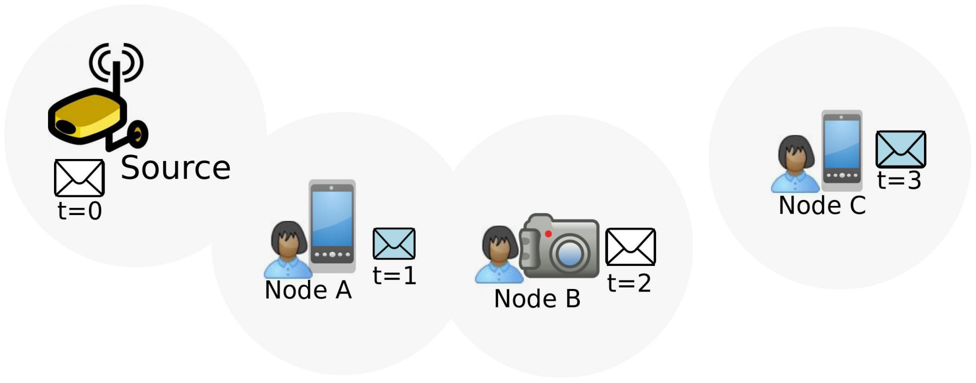


Figure 1: An intermittently connected Internet of things Multicast example. Different nodes containing application instances with different characteristics. In t=0 a message is created. Then, this message is forwarded in a store-carry-and-forward way from the source, to Node A, then to Node B and finally to Node C. The message is delivered to nodes A and C (darker message means delivery).

be the destination of application messages. When a node does not host any application instance, it merely forwards messages following a given routing protocol. In contrast to traditional Internet-like networks that require continuous end-to-end connectivity, every OppNet node is able to send, receive and forward messages.

Messages may be addressed to a single application instance or a group of destination application instances simultaneously, as depicted in Figure 1. As introduced by Carter et al. [8], Manycast is a communication scheme that allows a sender client communicate with an arbitrary number of group members.

In this article, Softwarecast, a general delivery scheme for Manycast group communications is presented. We define N as the set of all the nodes in the network. The number of nodes in a network is the cardinal of the set N ($|N|$). In the very same way, A is defined as the set of all of the application instances in a network. The number of application instance in a network is the cardinal of the set A ($|A|$). A node may run more than one application instance.

The *message state* is defined as a set of variables that defines the state of a message. Examples of message state variables could be the number of hosts a message has been forwarded to or the number of times the message can be replicated. We define S as all of the possible states of a message.

The *delivery condition* (c) is defined as a condition that allows the applications to define refined delivery expressions to select $|G_c|$ application instances from the A available at delivery time:

$$c : A \text{ x } S \rightarrow \{0, 1\}$$

A *delivery target group* (G_c) is defined as the set of application instances that satisfy a given condition c:

$$G_c = \{app \in A | c(app, s) = 1\}$$

These delivery conditions and the message states allow messages to be delivered in terms of refined strategies. Delivery decisions are made locally on every node in terms of the delivery conditions. However, delivery decisions that take into account not only information from the current node where the message is in custody, but from other nodes as well, can be accomplished using the message state. The combination of the *delivery condition* and the *message state* is extremely compelling: we can make the messages to behave, in terms of message delivery, as powerful and versatile as a finite state machine.

To better understand our delivery scheme, an example in the context of disconnected emergency scenarios is shown. The different communication application instances running on the portable devices belonging to the emergency medical team can be characterised in terms of their surgeon knowledge. Communication applications belonging to different emergency personnel will have different values for this characteristic: a high value for a surgeon and a low one for the rest of the doctors. A surgeon Manycast message may be defined as a message directed to the set of communication application instances in the emergency scenario intermittently disconnected network that are characterised as surgery experts:

$$c(app, s) = \begin{cases} 1 & if\ app.surgeonKnowledge() >= threshold \\ 0 & otherwise \end{cases}$$

Following the emergency scenario intermittently disconnected network example, a message could be addressed to the first k surgeons to be found, to all of them or the best k ones taking into account a certain criterion, like for example, its availability to come back to the Emergency Coordination Center. In this case, it will be very advantageous to allow the message to keep information, using the *message state*, such as for example the number of messages already delivered or the maximum local value seen according to the studied criterion.

This kind of delivery condition is much more complex than the previous ones enumerated in this section. In the next section, it will be discussed how to apply them to OppNet to improve Manycast delivery.

3.1. Softwarecast Messages

Given a source application instance in a source node (*source*) and a given payload message (*payload*), a delivery condition (c) and a message state (s), *Softwarecast messages* are defined as messages ($m(source, payload, c, s)$) sent to a set of $|G_c|$ members of the set G_c. However, these conditions are difficult to be included in messages using traditional OppNet protocols. In this paper, we propose to use mobile code to implement them. Other proposals such as the one proposed by Borrego et al. [6, 5] already include mobile code in their messages but for other purposes such as routing, lifetime control or prioritisation. In Section 6, we present how to give expression to Softwarecast messages in a real OppNet network. We will discuss now how Softwarecast can be seen as a generalisation of classic network delivery schemes.

- Unicast is a group communication scheme where messages are sent to a single network destination identified by a unique address. Softwarecast Unicast messages may be seen as messages $(m(source, payload, c, s))$ where the c condition identifies in a boolean way the single node:

$$node = app.thisNode()$$
$$c(app, s) = \begin{cases} 1 & if\ node == destination \\ 0 & otherwise \end{cases}$$

- Multicast is a group communication scheme where messages are addressed to a group of destination computers simultaneously. Multicast may be seen as Softwarecast message $(m(source, payload, c, s))$ where the delivery condition $c(app, s)$ selects every application instance from a set of subscribed application instances:

$$c(app, s) = \begin{cases} 1 & if\ isSubscribed(app, group) \\ 0 & otherwise \end{cases}$$

- Anycast is a communication scheme that allows a sender to send a message to any node that belongs to a group of nodes. An Anycast message can be seen as a Softwarecast message $(m(source, payload, c, s))$ where the delivery condition $c(app, s)$ selects the first found application from a group of application instances in a network. In this case, the delivery condition can be formalised as:

$$c(app, s) = \begin{cases} 1 & if\ belongs(app, group) \\ 0 & otherwise \end{cases}$$

- A Broadcast message is a message sent to all possible recipients simultaneously. A Broadcast message may be seen as Softwarecast message $(m(source, payload, c, s))$ where the delivery condition $c(app, s)$ selects every application instance from a subgroup of size $|A|$, that is, every application instance in the network.

$$c(app, s) = true$$

- A Geocast message is a special form of Multicast where the possible recipients of a message are identified by their geographical locations. A Geocast message may be seen as a Softwarecast message $(m(source, payload, c, s))$ where the delivery function $c(app, s)$ selects a group of application instances from a group of size m identified by a geographical location. A geographic destination address may be expressed in terms of any geometric figure that defines the delivery area.

$$c(app, s) = \begin{cases} 1 & if app \in delivery\_area \\ 0 & otherwise \end{cases}$$

3.2. Delivery Software code

We have seen in the previous sections, generic conditions that allow applications in OppNet to define refined delivery expressions to select application instances at delivery time. We propose to include these conditions within the Softwarecast messages as software mobile codes. Softwarecast messages are not delivered in terms of their destination addresses. Instead, they are delivered in terms of these mobile codes. When a node has accepted custody of a Softwarecast message, the node executes its delivery software code for every application willing to receive Softwarecast messages.

These delivery software codes are created by including the delivery condition presented in this section as the condition to deliver the message. In Algorithm 1, a generic delivery software code is presented. Details on how messages are extended and how mobile codes are executed can be found in Section 6.

Algorithm 1 Generic Delivery Software Code

```
Ensure: $app$ is an application considered for message delivery
Ensure: $s$ is the set of variables used by the delivery conditions to keep their state
Ensure: $c(app, s)$ is a delivery condition
1: if $c(app, s)$ then
2:     deliver(app)
3: end if
```

3.3. Summary

In this section, we have seen a general delivery scheme that allows new communication schemes beyond the traditional Manycast paradigms. Softwarecast allows the applications to flexibly and accurately define which $|G_c|$ application instances should be the message recipients in terms of real-time, application-defined, context-aware or history-based conditions.

As a consequence of this flexibility, novel delivery schemes appear to be possible, and this opens up a new way for applications to communicate. This new paradigm allows defining delivery conditions that follow inter-application criteria: an application instance can be selected as part of the delivery target group in terms of a condition that takes into account other information from other applications. The result are Manycast messages delivered to a group of application instances that may use operations such as *k-first*, *average*, *maximum* or *minimum*, for example.

These delivery decisions are taken after messages have been received. Therefore, nodes that are willing to receive messages only if these messages can be delivered to a local application must, regrettably, accept all Softwarecast messages, even if afterwards these messages are not locally delivered. Dealing with selfish nodes that do not accept messages for forwarding in OppNet is not a trivial matter. Incentive schemes like the ones presented by Sánchez-Carmona et al. [30], that promote the cooperation of OppNet nodes can be very advantageous in these situations.

However, there are many more scenarios in which Softwarecast can be advantageous beyond those with a theoretical or statistical solutions mentioned below. For example, in some OppNet scenarios, available destination addresses are not always known by sending applications. To solve this problem, profilecast delivery schemes [11] allow message destinations as groups of users defined by their profiles. These profiles are very effective ways of characterising nodes in terms of node's attributes. Nonetheless, in certain scenarios, Softwarecast can improve profile-casting by allowing applications to send messages to a subgroup of these members. For example, a message could be sent to a certain profile filtered by those having *high* values for a certain characteristic. A sender application may not know in advance which are *high* or *low* values for this characteristic for the given profile. In this delivery problem, the delivery software code along with the message state can be advantageous to implement such a complex delivery action.

4. Practical application

In this section, we propose an open OppNet problem and its solution to illustrate the benefits of using the Softwarecast proposal introduced in the previous section. This is a real, representative but not unique example to show the limitations of current Manycast protocols in OppNet and how Softwarecast is an excellent solution to solve this kind of problems efficiently. We first explain a general open problem, the max-k-node delivery problem, then we propose a solution for this problem for a concrete case, and finally we show how to use Softwarecast to solve it.

The problem is a special case of Manycast where an OppNet application wants to send a message to k and only k application instances of an OppNet scenario that fit best a given criterion. This delivery problem is very difficult to be conducted using traditional proposals and may be advantageous in many different OppNet scenarios. For instance, in emergency scenarios, such as the ones presented by Borrego et al. [6], where the Emergency Coordination Centre could need to make a call to the best k disconnected doctors located in the emergency area that maximise a given criterion. This criterion can be factored out in terms of the doctor's characteristics such as its skills on some medical topic, problem or speciality. In emergency scenarios, as explained by Martín-Campillo et al [22] and Martí et al. [21], the time the medical stay in the emergency area before coming back to the Emergency Coordination Centre is a very delicate variable. This is why is very important that this Manycast message will be delivered to not more than k doctors because, otherwise, additional unwanted doctors will return to the Emergency Coordination Centre unnecessarily.

Another example could be in the context of the *consumer-buyer behaviour*. A client wants to buy k items from a group of $|N|$ buyers which have a product each with price $prize_{i,j}$ and are located in a disconnected OppNet scenario. The client wants to minimise the prize of their k purchases but does not know in advance the prize of the different offers from the different $|N|$ sellers from the disconnected scenario. The goal of this delivery method is to send a *buy message* in order to minimise the total cost of purchase $\sum_{i=1}^{k} prize_i$.

When these applications face these complex delivery challenges, traditional group communication schemes such as Broadcast, Multicast or Unicast fail to provide optimal strategies. Using Broadcast primitives to query the $|N|$ possible destinations will make the application receive only, because of the peculiarity of OppNet, responses from a subset of N. When being forwarded to this subset of nodes using Unicast messages, all of these messages may not arrive either, for the same reasons. We propose, in this section, to use search theory statistical techniques that guarantee optimal strategies to solve this type of complex delivery challenges. These strategies do not reach optimal decisions that can only be achieved when having a complete knowledge of the network, something extremely difficult in OppNet. However, we will see in Section 5 that these optimal statistical strategies performance is better than the ones based on traditional network primitives.

In this section, we bring to the fore the necessity of new group delivery schemes in OppNet and we explain how to solve this problem from a statistical and search theory perspective. For the sake of simplicity, from now on, we consider nodes with a single application instance. Then, we define the delivery condition as:

$$c : N \text{ x } S \rightarrow \{0, 1\}$$

In Section 6, we will discuss how to deal with more than one application instance per node.

4.1. The max-k-node delivery problem

The basic *max-k-node delivery problem* that will be discussed in this article can be stated formally as:

As introduced in Section 3, let N be the set of all the nodes in an OppNet network:

$$N = \{n_1, n_2, ..., n_{|N|}\}.$$

The number of nodes in a network is the cardinal of the set N ($|N|$). This number $|N|$ is finite and known. Given a node n_i, we define $r(n_i)$ as a criterion to evaluate a node. A node n_i is *better* than a node n_j, according to r, if:

$$r(n_i) > r(n_j)$$

For this problem, we consider the case were this criterion remains constant. This means that any given node n_i will always have a constant $r(n_i)$.

The n nodes in a network, in a hypothetical situation when seen altogether, may be ordered from best to worst unambiguously in terms of a certain criterion r. Moreover, we define the criterion of order of a set of nodes $P \subseteq N$ as:

$$r(P) = \sum_{n \in P} r(n)$$

Moreover, we define $\mathcal{P}$ as the set of all of the possible subsets of N:

$$\mathcal{P} = \{\{n_{i_1}, n_{i_2}, ..., n_{i_r}\} \subseteq N \setminus \emptyset\}$$

Additionally, let $\mathcal{P}_k \subseteq P$ be the set containing all the subset of $\mathcal{P}$ of size k:

$$\mathcal{P}_k = \{P \in \mathcal{P} | \, |P| = k\}$$

Note that:

$$|\mathcal{P}_k| = \big({}^{|N|}_{\;k} \big)$$

A message is forwarded from a source to different nodes until it is delivered to k nodes from the set N using opportunistic contacts. When a message is received, it is either delivered to a local application instance, forwarded to another node, or both. Let $C = \{n_{i_1}, n_{i_2}, \cdots, n_{i_t}\} \in \mathcal{P}$ be the set of nodes a message has been forwarded to and let $n_{i_{t+1}}$ a new forwarded node for potential delivery. The delivery condition, c, to decide whether this message should be delivered or not to $n_{i_{t+1}}$ is based only on the node and the previously forwarded nodes:

$$c : N \text{ x } P \rightarrow \{0, 1\}$$

In the max-k-node delivery problem a node wants to send a message to k and only k nodes $P = \{n_{i_1}, n_{i_2}, \cdots, n_{i_k}\} \in \mathcal{P}_k$ that maximises $r(P)$. A solution to the max-k-node delivery problem has to define a strategy to maximise the probability of selecting a set in $\mathcal{P}_k$, $\mathcal{O}$, such as:

$$\mathcal{O} = \{P \in \mathcal{P}_k \; | r(P) \geq r(P') \; \forall P' \in \mathcal{P}_k\}$$

This is equivalent to maximise a win function $\mathcal{W}$ that returns 1 if the k nodes that fit best to targeted criteria are selected and 0 otherwise. Given a list of nodes $P = \{n_{i_1}, n_{i_2}, \cdots, n_{i_k}\} \in \mathcal{P}_k$ a message has been delivered to, we define $\mathcal{W}(P)$:

$$\mathcal{W}(P) = \begin{cases} 1 & P \in \mathcal{O} \\ 0 & otherwise \end{cases}$$

There are different variations of the same problem that include scenarios where the maximum number of nodes is unknown, scenarios where without knowing the maximum number of nodes we do know the movement model distribution in advance and others with a known node arrival distribution. The goal of this section is to show that this problem may be solved using the Softwarecast proposal introduced in the previous section, not to implement all of the variations nor to find the best one for a given scenario.

4.2. Optimal Delivery Method

The optimal delivery method for the max-k-node delivery problem may be approached from a more analytical perspective. In statistics, in the context of search theory, there is the problem of choosing the best moment to take a particular action, in order to maximise an expected reward or minimise an expected cost.

In our max-k-node delivery problem, the decision is, when a message is forwarded to several nodes, when it should be delivered to the upper application. If the message is delivered to the first k encountered nodes it is quite likely that these nodes will not be the k best nodes in terms of the studied criterion. In the same way, if we wait for delivering the message to the last k forwarded nodes, it will be likely that we will discard the k best ones. However, the method of deciding after encountering every node in the network is not feasible: nodes may not be re-encountered again anymore. Consequently, the max-k-node delivery problem is a search theory decision problem, a problem of choosing the best moment to take the delivery action after encountering several nodes in order to maximise the given criterion.

4.2.1. Statistical solution for the max-k-node delivery problem

In order to understand the max-k-node delivery problem, we start by solving the max-k-node delivery problem for k=1. In this case, it is equivalent to the Marriage problem, explained by Freeman et al. [17][1], a well-known problem in applied statistics. A person is willing to marry a person out of $|N|$ rankable in terms of a given criterion. The person dates different marriage candidates and decides about whether to ask them to marry or not. Once rejected a candidate he/she cannot be dated again. During the relationship, the person can rank the marriage candidate among all candidates dated so far, but is unaware of yet unseen marriage candidates. The question is about the optimal delivery method to maximise the probability of selecting the best marriage candidate. The difficulty of this type of problems is that the decision must be made while dating the candidate, similar to the node encounters in the max-k-node delivery problem.

We apply to our OppNet scenario the search theory to solve the max-k-node delivery problem for $k = 1$. Following the statistical approach described by Freeman et al. [17], when we apply it to OppNet, we define v, as the number of nodes a message so far has been forwarded to and s, the apparent rank of the v_{th} last forwarded node. At any time of the max-k-node delivery problem for $k = 1$, the state of this process is described by $(v,\, s)$. If $s \neq 1$ there is certainly no interest in delivering the message to the v_{th} node as it cannot possibly be the best. After the next node has been contacted, the new state of the process will be $(v + 1, s')$, where s' is equally likely to be any one of the values $1, 2, ..., v+1$. If $s = 1$, this node is a candidate for delivery. The probability that the node is the best of all $|N|$ nodes is just $v/|N|$. Letting $P(v, s)$ denote the maximum expected probability of choosing the best node to deliver the message when the state of the process is (v, s), the principle of dynamic programming yields the equations:

[1] The marriage problem is also known as the secretary problem.

$$P(v,1) = max\{\frac{v}{|N|}, \frac{1}{v+1}\sum_{s'=1}^{r} P(v+1,s')\}$$

$$P(v,s) = \frac{1}{v+1}\sum_{s'=1}^{r} V(v+1,s')(s = 2,3,...,v)$$

with $P(n,s) = 1$ if $s = 1$ and 0 otherwise.

Lindley et al. [20] solved this problem by simple backwards recursion over $v = |N|, |N| - 1, ..., 1$:

$$a_v = \frac{1}{v} + \frac{1}{v+1} + ... + \frac{1}{|N|-1},$$

The optimal action in state $(v,1)$ (when it is clear that the v_{th} last forwarded node is the best) is to deliver the message if $a_v < 1$. Instead, if $a_v > 1$, the optimal action is to continue, which means doing a simple routing. If v^* is defined as the integer v for which $a_{v-1} >= 1 > a_v$, the optimal delivery method for the max-k-node delivery problem for $k = 1$ is to reject the first $v^* - 1$ nodes and then to deliver the message to the first node thereafter that is better than all previous ones. The probability of winning using this policy is $(v^* - 1)a_{v^*-1}/|N|$. As $|N| \to \infty$, both this and $v^*/|N| \to e^-1 = 0.368$.

In a more general way, when we want to give a solution for the max-k-node delivery problem for a general k, we apply the generalization of the best choice problem proposed by Nikolaev et al. [27]. The solution to the max-k-node delivery problem for a general k is a strategy where there exists an optimal set $v_1^*, v_1^*, v_2^*, ..., v_k^*, 1 \leq v_1^* \leq v_2^* \leq \cdots \leq v_k^* \leq |N|$ such that the pth copy of the message ($p \leq k$) is delivered:

- to the first node which is the best of all the previous ones, if the message has been forwarded to v_1^* nodes.
- or, if not found, to the first node which is the second best of all the previous ones, if the message has been forwarded to v_2^* nodes.
- or, if not found, to the first node which is the third best of all the previous ones, if the message has been forwarded to v_3^* nodes.
- etc.
- or, if not found, to the first node which is the kth best of all the previous ones, if the message has been forwarded to v_k^* nodes.
- or, if not found, to the $(|N| - k + p)th$ node.

In other words, the solution to the max-k-node delivery problem, for any k, is a complex strategy. The Softwarecast delivery scheme is capable of implementing such complex strategies as a result of combining software code and message state. Instead, traditional Manycast protocols are not capable of implementing such complex strategies and, as a consequence, they are not able to solve, using optimal strategies, complex delivery problems such as the max-k-node delivery problem or others presented in Section 3.

4.3. Manycast Problem $k = 2$

We study a special case of the max-k-node delivery problem: the max-k-node delivery problem where $k = 2$. This is a case of Multicasting where messages are sent to two concrete nodes in an OppNet, those which fit best a certain criterion. We consider a win event if the message is delivered to the best and second best node from a group of $|N|$ nodes. We follow the optimal strategy explained in the previous section introduced by Nikolaev et al. [27]. Firstly, skip the first $v_1^* - 1$ nodes and then deliver the first copy of the message to the first node which is better than all of the previous ones, or to the $(|N| - 1)th$ node, if this node does not appear. Secondly, deliver the second copy of the message to the first node which is better than all of the previous ones or, if the message has been forwarded to more than $v_2^* - 1$ nodes, to the second best from all of the previous ones, or to the $|N|\, th$ node, if these nodes do not appear.

For this problem, for $k = 2$, Tamaki et al. [36] propose the values for v_1^* and v_2^* as $0.2291/|N|$ and $0.6065/|N|$ respectively. In the experimentation section (Section 5) we will study the performance of Softwarecast when solving this special case of the max-k-node delivery problem for these values.

4.4. Softwarecast Approach for solving the max-k-node delivery problem

From the communication point of view, the solution to the max-k-node delivery problem is a Manycast message that has as destination address a group of two application instances hosted in two different nodes. When it comes to solving complex Manycast problems such as the max-k-node delivery problem, traditional Manycast OppNet protocols fail to give complete solutions. In other words, the delivery of these messages cannot be directly achieved using traditional Manycast delivery schemes such as Multicast or Broadcast: there is no way of expressing such a refined destination using the traditional group delivery schemes.

We propose to solve the max-k-node delivery problem using a Softwarecast message $(m(source, payload, c, s))$ where the c condition represents the optimal search theory delivery method.

For the max-k-node delivery problem, for any given k, the solution is a message $(m(source, payload, c, s))$ where c is:

$$c(node, s) = \begin{cases} 1 & (l \;>\; max_1 \;\;\&\;\; forwarded > v_1) \;\; || \\ & (l \;>\; max_2 \;\;\&\;\; forwarded > v_2) \;\; || \\ & (l \;>\; max_3 \;\;\&\;\; forwarded > v_3) \;\; || \\ & ... \\ & (l \;>\; max_k \;\;\&\;\; forwarded > v_k) \;\; || \\ & (forwarded == (|N| - k + p)) \\ 0 & otherwise \end{cases}$$

where p, max_i, $forwarded$, and v_i^* are *message state variables*, as defined in Section 3, that represent the number of the message to be delivered (first, second, ..., kth), the *ith* best value for the studied local characteristic, the number of nodes the message has been forwarded to and the optimal value for optimal delivery, respectively. Additionally, l is the studied local value retrieved from the node.

In Algorithm 2, an algorithmic design for the strategy for the *max-k-node delivery problem* for *k=2* is presented. Additionally, a concrete deliver software code for the *max-k-node delivery problem* will be shown in Section 6.1.

Algorithm 2 Algorithmic design for the *max-k-node delivery problem* for *k=2*.

```
Ensure: forwarded is the number of nodes the message has been forwarded to
Ensure: k is the number of copies of the message to deliver
Ensure: localnode is the local node
Ensure: p is the number of the message copy to be delivered (first, second, ..., kth)
Ensure: l is the value for the studied local characteristic
Ensure: $max_1$ and $max_2$ are best and second best values for the studied local characteristic
Ensure: $v_i^*$ is the ith optimal value for optimal delivery
#Initialise variables
k=2
forwarded=0
p = 1
$max_1 = 0$
$max_2 = 0$
#Loop while the message has not been delivered to k nodes
while $p \leq k$ do
    $forwarded++$
    #Check if message must be delivered
    for $i = 1$, $i \leq k$, $i$++ do
        if $(l \geq max_i \ \& \ forwarded > v_i^*) \ ||(forwarded > (|N| - k + p))$ then
            deliver(localnode)
            p ++
            break
        end if
    end for
    #Update $max_1$ and $max_2$
    if $l > max_1$ then
        $max_1 = l$
        $max_2 = max_1$
    else
        if $l > max_2$ then
            $max_2 = l$
        end if
    end if
    #The message must still be delivered elsewhere. Route it.
    route()
end while
#The message has been delivered k times. Remove it.
delete()
```

In this section, we have described a use case as an example that illustrates very clearly the potential of our proposal. In the following sections, we analyse the performance of our proposal when solving this use case and we propose a way of giving expression to Softwarecast messages in a real OppNet.

5. Simulation Results

In this section, we present a series of simulations to study the performance of our solution for the max-k-node delivery problem for $k = 2$ presented in Section 4. We show, by means of these simulations, how effective it is to allow messages to carry software codes to make the delivery decisions. We have chosen $k = 2$ as an illustrative Manycast example. Other values of k can easily be achieved in a very same way. As we have seen in Section 4.4, there is a general Softwarecast approach for solving the max-k-node delivery problem for any given k. The difference in terms of the delivery software code size for the max-k-node delivery problem for $k = 2$ and a bigger k is not significant, even less in many cases, like the one presented in this section, where the delivery software code represents less than 1% of the total message size. Additionally, this difference regarding the size of the message state is totally insignificant: the message state would increase in just a few bytes for bigger k because, as introduced in Section 4.2.1, k values with the k best values for the studied local characteristic must be kept within the message state.

The simulations in this section has been conducted using a modified version[2] of the Opportunistic Network Environment (TheONE) simulator [14]. This simulator was originally created to run simulation experiments on opportunistic networks like DTN and to provide reports and graphical representations of the results. TheONE represents the communication among nodes and their movement. We have modified TheONE to be able to simulate Softwarecast messages presented in Section 3 as well as other Manycast approaches in order to study our network proposal performance. Additional TheOne reports have been programmed to study variables such as the *win ratio*, explained in Section 4.

5.1. Settings

In these simulations, we consider a group of different nodes that form a communication network similar in characteristics as the one proposed by Borrego et al. [6] to make it more realistic. In an area of 4 km^2 located in the surroundings of the Autonomous University of Barcelona[3], different nodes with two different group of nodes share the network, group A and group B. The buffer size for the nodes is fixed to 2GB[4]. Nodes carry a wireless interface controller operating in Ad Hoc mode with a transmit speed fixed to 54Mbs. Maximum range obtained is 150m. There are 40 nodes of group A and 30 of group B. Group B nodes act as simple message forwarders. Messages created

[2]The source code for this TheOne modification can be found at https://senda.uab.cat/wiki/aDTN in Section "TheOne resources".

[3]Map used: http://www.openstreetmap.org/node/259644263#map=15/41.5039/2.0825

[4]Having a MicroSDHC from a Raspberry Pi as a reference.

during the simulations have a variable size from 50KB to 100KB[5]. Messages are created periodically using a random value between 1 and 10 seconds and are routed using a single-copy routing protocol [28]. Nodes move inside the simulation area according to the map previously mentioned at a maximum speed of 1.4 m/s following a random waypoint movement model [29]. Several points of interest have been defined to make the movement more realistic[6]. When Softwarecast extensions are present in the message, the size of these extensions is 400 bytes. All simulations represent 24 hours of activity. The goal of these simulations is to understand the performance of Softwarecast messages sent from a single node in a network to nodes of group A in a period of time of 3 hours[7]. Every simulation is repeated 100 times with different random seeds. These random seeds define the random component of the node's movement model.

5.2. Scenario study

In opportunistic networks, the maximum number of unique nodes a Softwarecast message can be forwarded to is difficult to know. We have performed 100 different simulations with different movement random seeds in order to understand the maximum number of unique nodes a Manycast message is likely to be forwarded to in the studied scenario. We want to understand if it is easy to find $|N|$, that is, the maximum number of nodes a Softwarecast message will be able to be forwarded in a fixed period of time of 3 hours.

In Figure 3(a), the inter-unique-message-node-contact time for a period of 3 hours is studied for these different random seeds. This represents the time, on average, between contacts of messages with nodes that have never been forwarded to before. It can be seen that this time for the different random seeds is very variable, as seen by its standard deviation. Additionally, in Figure 3(b), for the same simulations, we see an aggregated view of the frequency of the total number of contacts after 3 hours. We see that the average is 17.9, but again, there is quite a big variability. Finally, in Figure 4(a), we see the maximum number of unique nodes a Broadcast message is forwarded to in the same period. The average is 17.9, but we see that there is a wide number of different values. We understand by this three figures that choosing an $|N|$ for this scenario is not an easy decision.

In Figure 4(b), we see the result of 9175 simulations for the win ratio in terms of the different values of v_1^*, v_2^* and $|N|$ where $2 <= v_1^* < v_2^* < |N| <= 40$ (40 is the number of nodes in group A). We see that the maximum win ratio is obtained with the values $v_1^* = 4, v_2^* = 8$ and $|N| = 38$ obtaining a win ratio of 0.63. This combination of v_1^*, v_2^* and $|N|$ is very difficult to foresee, according to the variable statistics presented

[5]This size is representative of messages used in emergency scenarios as learnt by our research group from projects like the ones presented by Martín-Campillo et al. [23] and Martí et al. [21] and some others from other external studies [19, 7].

[6]P1: 41.5008155, 2.0905975, P2: 41.50361, 2.0938, P3: 41.50392, 2.08922, P4: 41.50851, 2.08998, P5: 41.50508, 2.08402 P6: 41.49975, 2.08078

[7]Half the time the paramedic personnel stays in the emergency area before coming back to the Emergency Coordination Centre.

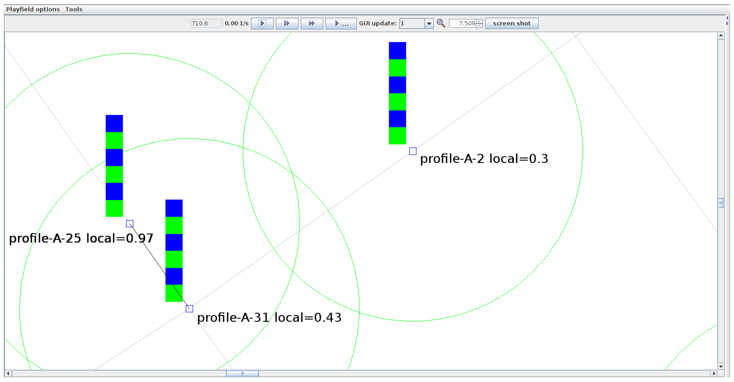
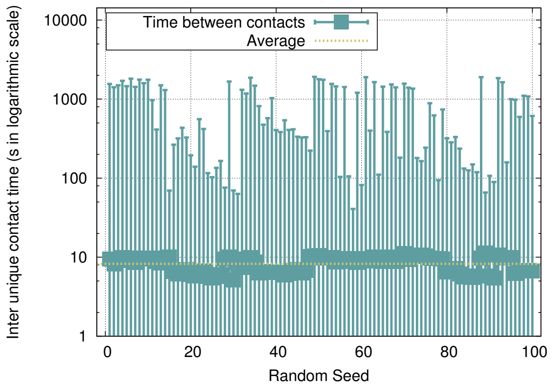
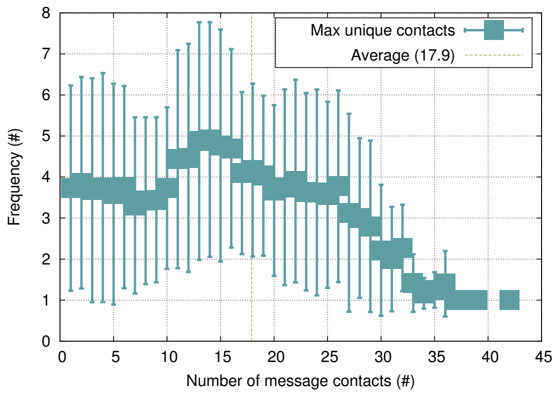


Figure 2: In this figure, a snapshot from a simulation is depicted. Three nodes with different local values for a studied characteristic are shown. The two nodes in the left part of the figure are in contact. A message m is forwarded from node *profile-A-2* to node *profile-A-31* and to node *profile-A-25*. The message is delivered to this latter according to the delivery condition in m.



(a) Inter-unique-message-node-contact time in seconds as a function of the different random seeds for the 100 different simulations. Global average for all simulations is 8.24 seconds.



(b) Distribution of the number of unique nodes a node is forwarded to during the simulation.

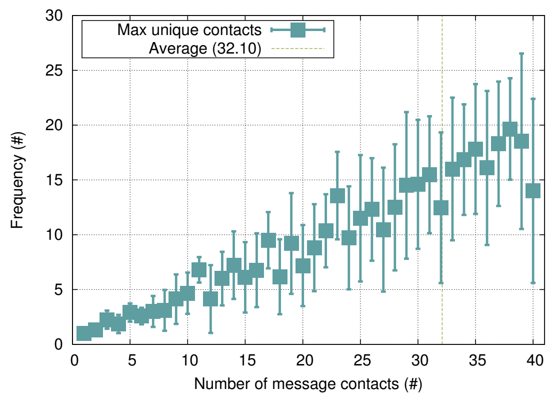
Figure 3: Study of the movement model behaviour: inter-unique-message-node-contact time and distribution of the number of unique nodes a node is forwarded to.

in figures 3(a), 3(b) and 4(a). We choose for future simulations a compromise value for $|N|$ as being 40, the maximum number of nodes of group A. As a consequence, following the statistical model presented in 4, values for v_1^* and v_2^* are $v_1^* = 0.2291 * |N| \sim 9$ and $v_2^* = 0.6065 * |N| \sim 25$. As we will see in this section, it gives an excellent result.

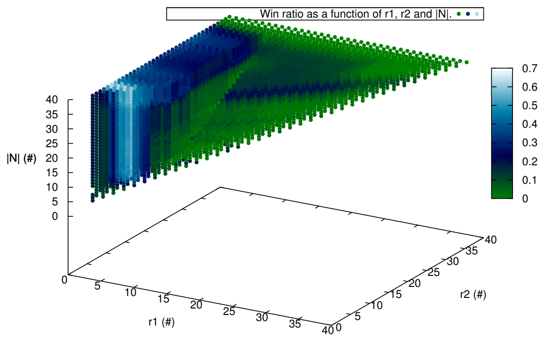
5.3. Comparing to traditional approaches

We compare in additional simulations, four ways of solving the max-k-node delivery problem for a period of 3 hours. This means we want to maximise the number of messages delivered to the best 2 nodes from a certain group in terms of a certain criterion in 3 hours time.

Firstly, we simulate the performance of our analytical proposal introduced in Section



(a) Distribution of the number of unique nodes a flooding message is forwarded to during the simulation.



(b) This figure represents 9175 simulations to analyse the win ratio as a function of v_1^*, v_2^* and $|N|$. We loop over the different values of v_1^* and v_2^* where $2 <= v_1^* < v_2^* < |N| <= 40$.

Figure 4: Study of the movement model behaviour: flooding message number of unique node contacts and win ratio as a function of v_1^*, v_2^* and $|N|$.

4, using $v_1^* = 9$, $v_2^* = 25$ and $|N| = 40$ (SC-Softwarecast in the figures). As explained in Section 4.3, the strategy to solve the max-k-node delivery problem is to skip the first 8 nodes and then deliver the first copy of the message to the first node which is better than all of the previous ones, or to the $39th$, if this node does not appear. Secondly, deliver the second copy of the message to the first node which is better than all of the previous ones or, if the message has been forwarded to more than 24 nodes, to the second best from all of the previous ones, or to the $40th$, if these nodes do not appear.

Secondly, we have compared our Softwarecast proposal with a multi-copy version of Softwarecast (MC-Softwarecast in the figures). In this case, messages are not being routed following single-copy routing algorithm but a multi-copy one similar to Spray and Wait [13] in binary mode: messages define an initial number of copies ($L = 2$) and, when being forwarded to other nodes, half of their Softwarecast messages are forwarded to the contacted node (one copy since $L = 2$). These multi-copy Softwarecast messages, when being forwarded, follow a strategy similar to the Marriage Problem [16] explained in Section 4 with $v_1^* = 40/e = 14.7$: messages are not considered for delivery before being forwarded to 15 nodes and after the 15th contacted node the message is delivered to the first node whose retrieved value is bigger than all of the precedents. We will see in the following simulations that the multi-copy approach performs worse than the single-copy because, in a not insignificant number of times, both copies of the Softwarecast message select the same node for message delivery.

Thirdly, we analyse the performance of a classic approach of solving the max-k-node delivery problem. Since traditional Manycast can not be directly applied to this problem since nodes are not capable of knowing the value for the studied criterion for other nodes, we compare our proposal with another that uses traditional up-to-date network primitives. In this traditional proposal, the sender sends queries to the different nodes belonging to the given group to retrieve the value of the studied criterion. These queries are performed using EBP [2], an efficient Broadcast protocol that retransmits a message only when the forwarder is surrounded by at least a certain number of neighbours and if at least one of these neighbours has not received the message yet. Afterwards, the sender chooses from the received messages the best 2 ones and sends a Unicast message to both of them. These Unicast messages and the query responses are routed using Spray and Wait [13]. We call this traditional approach *a two-step Broadcast-Unicast-based* delivery solution (*2sBcastUcast* in the figures).

Finally, we analyse the performance of an Oracle routing [37] where nodes have a perfect knowledge of the network. This perfect knowledge is impossible to be obtained in almost every OppNet scenario. However, we simulate it to understand which are the better results we can achieve for the studied scenario.

In Figure 5(a), the win ratio is studied. Three different delivery schemes are studied in this figure. The two-step Broadcast-Unicast-based and the Softwarecast are compared with a hypothetical Oracle-based delivery scheme. In this delivery scheme, messages have the perfect knowledge of the network in terms of the studied criterion and they are able to make perfect delivery decisions. As it can be seen, both Softwarecast delivery schemes, single-copy and multi-copy, perform better than the Broadcast-Unicast-based (0.35, 0.28 and 0.16 on average), but not as good as the maximum as seen from the Oracle-based (0.9 on average). In Figure 6(b), we analyse for the series of these simulations the number of messages delivered for a period of 3 hours on average after v_1^* contacts, after v_2^* contacts and when the number of unique nodes contacted is exactly the maximum number of existing nodes for the targeted destination. As expected, the first two delivery options join the 99% of the messages delivered (42,5% for the first one and 56,8% for the second one).

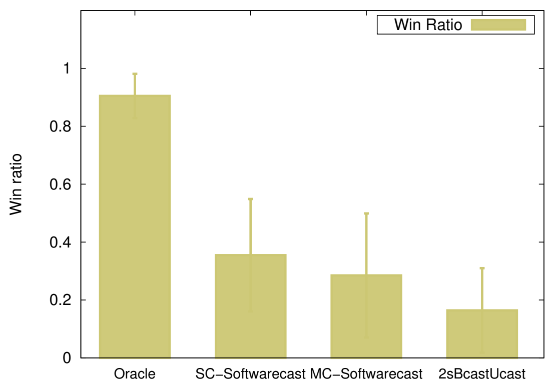
In Figure 5(b), the latency time is studied comparing the four different schemes: single-copy Softwarecast, multi-copy Softwarecast, Oracle and two-step Broadcast-Unicast-based. In this case, the latency for both Softwarecast, highly improves the two-step Broadcast-Unicast-based, and they very close to the optimal one, the Oracle delivery.

In Figure 6(a), the number of hops is studied for the same series of simulations. We see that both Softwarecast proposals employ less number of hops than the two-step Broadcast-Unicast-based, but still, is worse than the Oracle one.

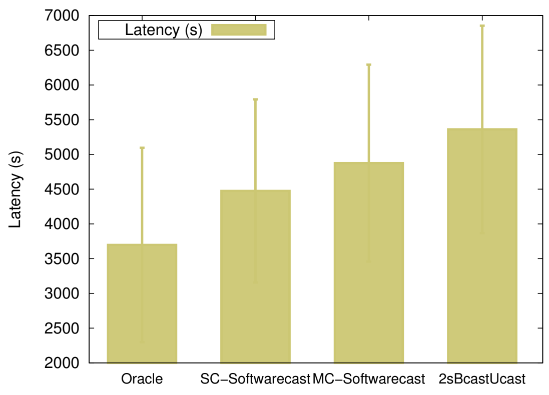
We have seen in this section, using simulations and based on a particular scenario, that allowing messages to carry software codes to make the delivery decisions can solve some OppNet problems like the max-k-node delivery problem. In the next section, we propose an implementation of our proposal to show its viability in a real OppNet network.

6. Softwarecast in the Bundle Protocol

In this section, we show the viability of our proposal, introduced in Section 3, by extending the Bundle Protocol (RFC 5050 [31]) in order to allow Softwarecast messages to be implemented in a real OppNet. We have chosen to design our implementation in the context of DTN networks ([9, 38]), a special type of OppNet that is based on the Bundle

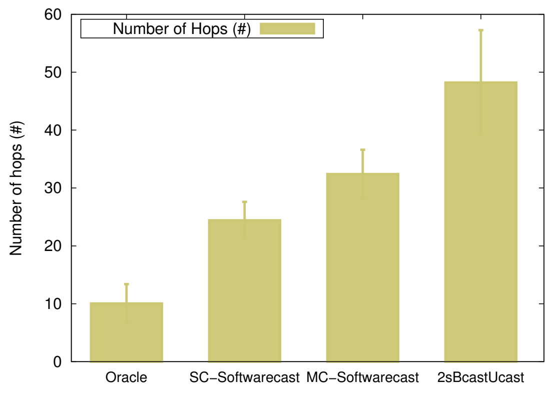


(a) Win ratio in (0,1) for the four group delivery schemes: Oracle-based, single-copy Softwarecast (SC-Softwarecast), multi-copy Softwarecast (MC-Softwarecast) and two-step Broadcast-Unicast-based (2sBcastUcast).

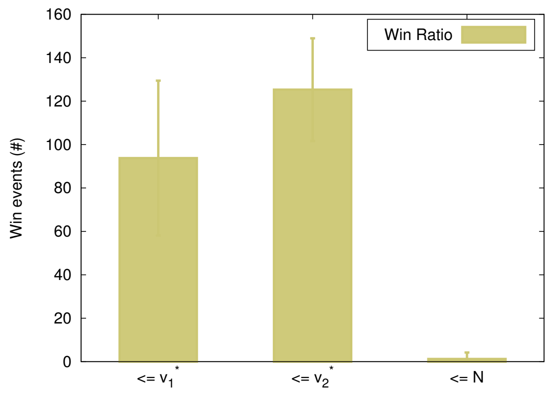


(b) Latency in seconds for the four group delivery schemes: Oracle-based, single-copy Softwarecast (SC-Softwarecast), multi-copy Softwarecast (MC-Softwarecast) and two-step Broadcast-Unicast-based (2sBcastUcast).

Figure 5: Performance studied by analysing the win ratio and the latency for four group delivery schemes: Oracle-based, single-copy Softwarecast (SC-Softwarecast), multi-copy Softwarecast (MC-Softwarecast) and two-step Broadcast-Unicast-based (2sBcastUcast).



(a) Number of hops per message for four group delivery schemes: Oracle-based, two-step Broadcast-Unicast-based and Softwarecast (multi-copy and single-copy).



(b) Win events to analyse the proportion of messages delivered during a period of three hours on average after v_1^* contacts, after v_2^* contacts and when the number of unique nodes is exactly $|N|$.

Figure 6: Performance studied by analysing the win ratio and the latency for four group delivery schemes: Oracle-based, two-step Broadcast-Unicast-based and Softwarecast (multi-copy and single-copy).

Protocol. By this, we show that our proposal may be implemented and so we prove its feasibility.

In the context of OppNet, a *DTN node* is a network node that implements the Bundle Protocol (BP) [31]. Nodes in the network may host application instances that create, or may be the destination of DTN messages (bundles) or simply messages. When a DTN node does not host any application instance, it merely forwards messages following a given routing protocol.

A Manycast message, as seen in section 3, contains a message source, its payload, and a delivery condition. We propose in this section, to implement delivery conditions using software mobile codes. Mobile code [34] is a well-known technology designed for transferring software code between systems and to be executed on a remote system without the need of an installation by the recipient.

For this purpose, we use Active-DTN [6], a DTN solution based on the Bundle Protocol where messages have an active routing behaviour. In Active-DTN, active messages are messages that beyond being data wrappers, they can carry software code for routing purposes. There are three basic software codes that can be included in a Active-DTN message : forwarding, lifetime control and priority. In this section, we explain how to extend this network paradigm with additional software codes to achieve Softwarecast Multicast delivery.

Active-DTN uses RFC 6258 [35], an extension block that may be used with the Bundle Protocol that defines how a message can be extended by means of adding a set of Metadata Extension Blocks (MEB). The purpose of MEBs, as defined in RFC 6258 [35], is to carry additional information that nodes can use to make processing decisions regarding messages. Active-DTN defines a Mobile code Metadata Extension Block (MMEB) as a type of MEB with the necessary fields for the inclusion of software code for routing purposes.

We propose to define an additional software code, the delivery software code, to include mechanisms to be able to implement the delivery conditions presented in Section 3. This software code will be then included in DTN messages as message extensions following RFC 6258 [35].

However, these software codes need a state that must be kept when being forwarded from one DTN node to another. This state represents historical information that allows the delivery software code to be capable of performing appropriately in new environments. Additionally, communication among forwarding and delivery software codes may be necessary to be able to implement all the Manycast delivery schemes presented in Section 3. For these purposes, we propose extending messages with a software code state that allows precisely this: persistent data storage and inter-software codes communication.

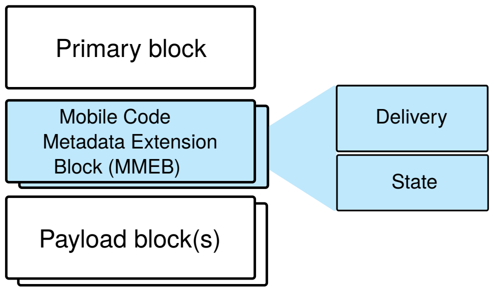


Figure 7: MEBs fit between the primary and the payload blocks of a Bundle message.

Our proposal is to extend Bundle messages with two types of blocks: delivery and software code state as depicted in Figure 7. These extension blocks are divided into additional fields following Active-DTN as described by Borrego et al. [6].

6.1. Implementation

In this section, we describe an implementation of the extensions described in Section 3. A first Active-DTN implementation was introduced by Borrego et al. [6]. In this section, we will focus on the Active-DTN details concerning the requirements to implement the Softwarecast messages in the context of the Bundle Protocol[8]. In Figure 8(a), an overview of the Active-DTN implementation is presented. Two different levels are depicted in this figure: the application level and the Bundle level.

The Bundle I/O manager is the module in charge of receiving Bundle messages from local application instances, through the Application Manager, or from other DTN nodes. When receiving a bundle message, the Bundle I/O manager executes, if found, its delivery software code for every local application instance willing to receive Softwarecast messages. Then, if the Bundle message requires being forwarded, it is enqueued in the Bundle Queue.

The Custody Manager dequeues Bundle messages when neighbours are discovered. For every Bundle message, it executes its forwarding software code, if any, to choose among the possible neighbours to be forwarded. If no forwarding software code is found, a default forwarding algorithm is applied.

The Application Manager is responsible for communicating with the application instances registered to the Active-DTN platform. Each application instance registers to the platform performing a connection through a local socket. Once registered, they can issue commands to the platform, for example indicating which Bundle messages want to receive, send Bundle messages, etc.

Additionally, application instances and devices, such as GPS or another type of sensors, once registered, can feed the Routing Information Table (RIT [6]), a place where software codes may read and write information that can be used for future forwarding or delivery decisions. Each application instance has its own table and also has access to a common table shared among all of them. For performance reasons, tables inside the RIT are implemented as hash tables. The RIT Manager module performs the entry cleaning in order to prevent permanent memory exhaustion problems.

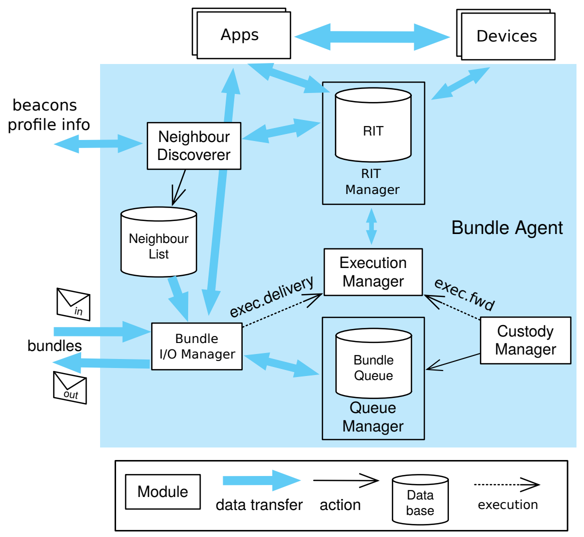
The Neighbour Discovery Module send beacons, as UDP Multicast datagrams, announcing the presence of the node and listens for beacons from other platforms. It maintains a neighbour table with information about the discovered neighbours which is used by the Custody Manager to send beacons to the nearby nodes.

The Execution Manager is responsible for executing the software codes carried by the messages. Following, we analyse the security aspects aDTN provides to guarantee security against malicious software codes.

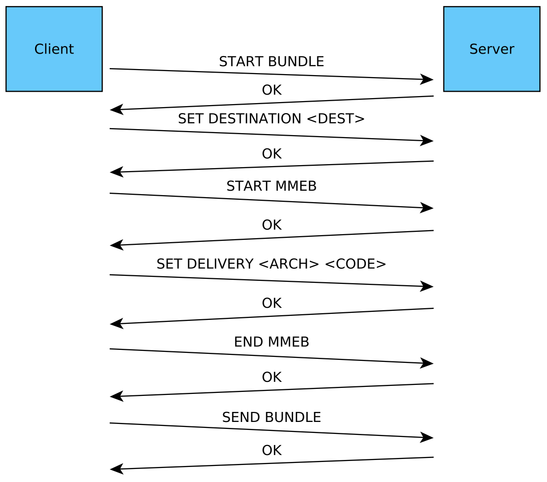
The software codes are executed in a virtual machine that interprets the eBPF instruction set[9], but support for other software code language formats is possible by extending the Execution Manager. For example, the first Active-DTN implementation presented by

[8]Source code for Softwarecast Active-DTN can be found at https://github.com/SeNDA-UAB/aDTN-platform/tree/softwarecast

[9]https://www.kernel.org/doc/Documentation/networking/filter.txt



(a) Structure of the Active-DTN platform.



(b) Example of a Bundle creation protocol: a Bundle is created with the forwarding MMEB and the delivery extension.

Figure 8: Bundle extensions.

Borrego et al. [6] used standard C code executed using TinyCC as compilation back-end, but we have found that we could improve the security of the execution while maintaining the performance and code size using eBPF software codes. A comparative between the two execution environments is shown later.

The eBPF architecture is an extension of the classical BPF (BSD Packet Filter [24]) instruction set with additional functionalities, like the ability to call external functions. It has been designed to be fast and secure, and currently it is being used by the Linux kernel to execute user-provided software code in kernel space for network packet filtering and tracing purposes.

The eBPF instruction set only allows to perform memory accesses within its own stack and registers, and to call a predefined set of external functions. These measures isolate the execution from the rest of the system. Additionally, before executing an eBPF program, the software code is statically analysed by the verifier to ensure that it can not compromise the system. The verifier simulates the execution of the program, bounds-checks all memory accesses and verifies that the software code will terminate by checking that the program has at most 4096 instructions with no infinite loops. Additionally, it ensures that all register operations are safe: it is not allowed to read a register before setting it and each register is tracked to make sure it always has the right type of data.

An additional layer of security is provided by executing the Execution Manager as a standalone process with a socket based interface, therefore it can be isolated in a container or in a virtual machine.

Fast execution is achieved by compiling the software code to native code before being executed (JIT compilation). By design, the eBPF instruction set maps to the architecture of modern CPUs so that the JIT compilation can be performed very fast. Once native

	Size	Compilation time	Execution time	Overall Time	Security
eBPF	496 B	54884 ns	360 ns	55244 ns	High
C	418 B	31625926 ns	180 ns	31626106 ns	Medium

Table 1: A comparison between C and ePBF for the delivery software code for Listing 1.

compiled, the code can be cached to avoid further recompilations. Software codes are easily written in C and then compiled to eBPF using the Clang compiler before being included in the message.

Figure 9(a) shows the size of the compiled eBPF code and the size of the resulting native compilation as a function of the uncompiled C code. Although it is difficult to know the time that would take a given processor to execute a certain number of instructions, as it highly depends on the processor architecture and the compiler optimizations, we can see, as depicted in Figure 9(b), that the cost of compiling code from eBPF to native bytecode and then executing it is in the order of hundred of thousands of instructions. As a reference, a Raspberry Pi[10] is capable of executing several hundreds of millions of instructions per second (MIPS).

In Listing 1, as an example, a delivery code for the *max-k-node delivery problem* for *k=1* is presented[11].

```
int forwarded = getState(0);
int max = getState(1);
int localvalue = ritGet("/localnode/value");
if (forwarded > 20 && localvalue > max) {
 deliverMsg();
 } else {
  setState(0, ++forwarded);
  setState(1, max(max,local));
 }
```

Listing 1: Example of a Software Code: A Delivery Code for the *max-k-node delivery problem* for *k=1*.

Following, we present a comparison between the previous execution environment, as introduced by Borrego et al. [6] and the new Execution Manager presented in this study. The results of this comparison are shown in Table 1. The compilation and execution times are measured in a desktop computer with a Xeon X5650 processor and 18GB of RAM. The results show that the overall execution time (compilation time + execution time) has improved with the new Execution Manager.

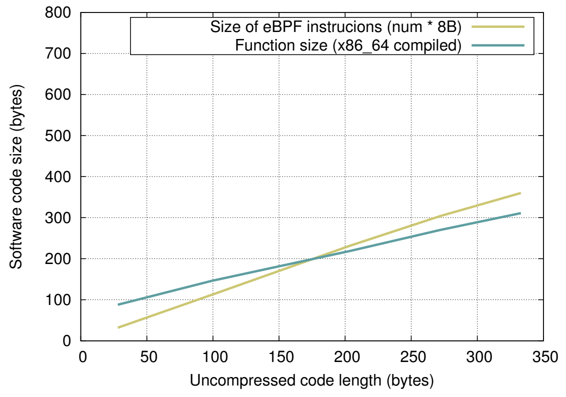
The current implementation has been coded in C language and provides a simple text-based communication API for application instances and routing algorithms to interact with the platform. In Figure 8(b) a sending example of a Bundle message with a delivery

[10]The Raspberry Pi is a low cost, credit-card sized popular computer.

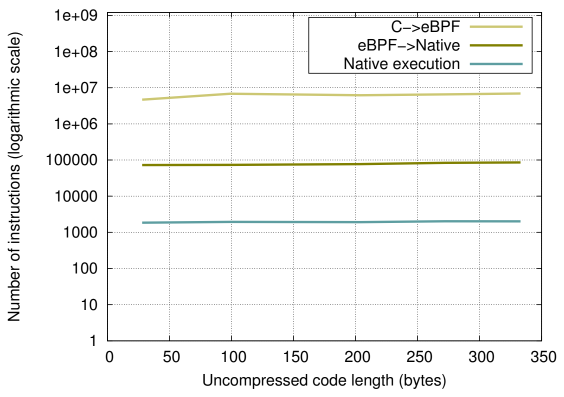
[11]The resulting code compiled to eBPF can be found at: `https://github.com/SeNDA-UAB/aDTN-platform/tree/softwarecast/swcodes`

code is depicted.

The software code state is initialized by adding a software code state Algorithm Type. Both forwarding and delivery algorithms may access and modify this content by using the *setState()* and *getState()* functions. Internally, data is stored in the last field from the MMEB block, the Routing Metadata field, as introduced in this section. Different variables may be stored, and its retrieval is possible without needing any additional length fields since the state variables are delimited using an end of variable character.



(a) Compiled eBPF and native code size as a function of uncompiled C code size.



(b) Number of instructions as a function of the uncompressed software code length.

Figure 9: Software code compilation performance.

6.2. Experimentation

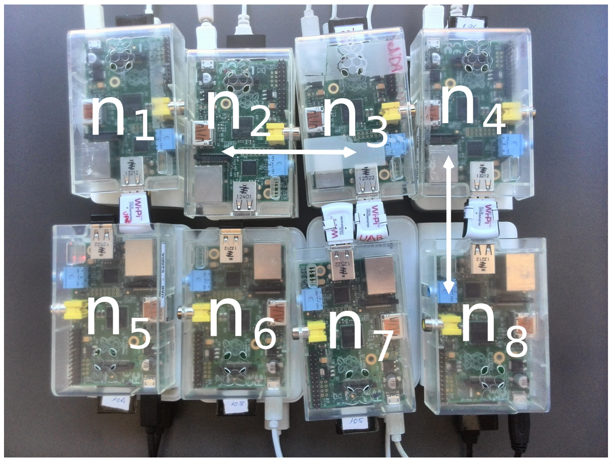
In this section, we present a real experiment to test the Active-DTN Softwarecast implementation in the Bundle Protocol. The objective of this experimentation is to study the performance of 100 Softwarecast messages that are sent to the 2 best nodes in the network according to a given criteria. Messages are delivered using the delivery software code presented in Algorithm 2.

The experimentation uses a network that consists of 8 Raspberry Pi[12] ($n_1 - n_8$). These nodes are ordered following a criteria r, such that $r(n_i) > r(n_j)$ if $i < j$.

In order to obtain intermittent connectivity among the nodes in the network, nodes are placed together, as depicted in Figure 10, and are disconnected from certain nodes, following a scheduled connectivity plan, using iptables[13]. Every 10 seconds the *INPUT*, *OUTPUT* and *FORWARD* chains from the 8 nodes are updated automatically. In the table from Figure 10, an example of the state of the network connectivity is shown.

[12]Raspberry Pi Broadcom BCM2835 SoC full HD, 700MHz Low Power ARM1176JZ-F, 512MB SDRAM, 4GB SD with Raspbian (Debian-based), equipped with a Wireless Edimax EW-7811Un (802.11b/g/n up to 150Mbps) with WMM-Power save mode, a GPS receiver NL-302U (baud rate: 4800 bauds, IPX6 protection class), NTP and a dual output 5000mAh battery.

[13]Administration tool for IPv4 packet filtering and NAT.



Node	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
n_1	1	0	0	0	0	0	0	0
n_2	0	1	1	0	0	0	0	0
n_3	0	1	1	0	0	0	0	0
n_4	0	0	0	1	0	0	0	1
n_5	0	0	0	0	1	0	0	0
n_6	0	0	0	0	0	1	0	0
n_7	0	0	0	0	0	0	1	0
n_8	0	0	0	1	0	0	0	1

Figure 10: Experimentation layout and state of the network connectivity example.

We have included in these 100 messages a delivery software code that implements the optimal stopping strategy described in Section 4.3. Messages are delivered using the delivery software code presented in Algorithm 2. Payload size for these messages was 10K. The delivery software code size was 400 bytes. From the 100 messages sent, 35% were optimally sent to nodes n_1 and n_2, the best ones according to criteria r. The average latency time for these messages was 83 minutes with a standard deviation of 10 minutes.

With the implementation described in Section 6.1, and the experimentation shown in Section 6.2, we show that it is feasible to define Softwarecast messages in Opportunistic Networks. From the implementation point of view, the most challenging part has been covering the security aspects for the fact of including software codes within the messages. We have found that it is possible to implement a fast and secure execution environment that fulfils the requirements of our proposal.

7. Conclusions

In this paper, we have presented a general delivery scheme based on mobile code to improve OppNet Manycast. This delivery scheme allows messages to carry software codes to make the application delivery decisions, that is, whether a message should be delivered to an application when being forwarded to a node.

Our proposal extends network addressing by moving from the static header field paradigm to a message-built-in software-code-based addressing scheme. These software codes may use a state also carried by the messages. The software code state allows to implement refined delivery behaviours so accurate delivery-decision-making methods can be performed. The result is a very powerful and versatile alternative to OppNet Manycast because delivery decisions can be made taking into account application-defined, context-aware or history-based conditions.

Following this delivery scheme and in order show how advantageous and powerful our proposal is, we have presented a statistical solution to the problem of sending a message to k and only k nodes of an OppNet scenario that fit best a given criterion. This problem, the max-k-node delivery problem, is mainly a delivery problem and it is very difficult to be conducted using its optimal delivery method by means of traditional network schemes like Manycast, Unicast or Broadcast. The max-k-node delivery problem is not the only problem that can be optimally solved. Our proposal opens new pathways for Manycast in OppNet.

Using a simulator capable of reproducing Softwarecast messages, we show that our proposal improves traditional ways of solving the max-k-node delivery problem for $k = 2$ in terms of win ratio and latency. We think this alternative can be advantageous in many OppNet cases like disconnected emergency scenarios or intermittently connected networks of heterogeneous physical objects. Specially when the network has a large number of heterogeneous nodes and the applications need dynamic and elaborate ways of selecting message destinations.

We present a down-to-earth implementation in the context of a real OppNet network to allow applications to send Softwarecast messages using the *de facto* standard for DTN, the Bundle Protocol. This implementation shows to be secure, performant, reliable, effective and powerful enough to reify our Manycast network delivery scheme.

As future lines of research, we plan to evaluate if sending first the delivery software code without the payload information could be beneficial for the network. When doing this, the payload message will only be forwarded if the result of the delivery software code is positive. However, we will have to evaluate how this will affect the performance of the network when, once decided that the message should be delivered, the contacted node with the payload information will not be available for routing anymore.

Acknowledgements

This work was partly supported by the Catalan AGAUR 2014SGR-691 project and the Spanish Ministry of Science and Innovation TIN2014-55243-P project. The authors thank Cristina Fernández for her generous help with the mathematical model and the anonymous reviewers for their careful reading of the original manuscript and their many insightful comments and suggestions.

References

[1] Mikael Asplund and Simin Nadjm-Tehrani. A partition-tolerant manycast algorithm for disaster area networks. In *Reliable Distributed Systems, 2009. SRDS'09. 28th IEEE International Symposium on*, pages 156–165. IEEE, 2009.

[2] Wafa Badreddine, Claude Chaudet, Federico Petruzzi, and Maria Potop-Butucaru. Broadcast strategies in wireless body area networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 83–90. ACM, 2015.

[3] Carlos Borrego, Sergio Castillo, and Sergi Robles. Striving for sensing: Taming your mobile code to share a robot sensor network. *Information Sciences*, 2014.

[4] Carlos Borrego and Sergi Robles. Relative information in grid information service and grid monitoring using mobile agents. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, pages 150–158. Springer, 2009.

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

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

[7] Raffaele Bruno, Marco Conti, and Andrea Passarella. Opportunistic networking overlays for ICT services in crisis management. In *Proceedings of International Conference on Information Systems for Crisis Response and Management ISCRAM*, 2008.

[8] Casey Carter, Seung Yi, Prashant Ratanchandani, and Robin Kravets. Manycast: Exploring the space between anycast and multicast in ad hoc networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 273–285. ACM, 2003.

[9] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. RFC 4838, delay-tolerant networking architecture. RFC 4838 (Informational), 2007.

[10] 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.

[11] Xia Deng, Le Chang, Jun Tao, Jianping Pan, and Jianxin Wang. Social profile-based multicast routing scheme for delay-tolerant networks. In *Communications (ICC), 2013 IEEE International Conference on*, pages 1857–1861. IEEE, 2013.

[12] Mieso K Denko. *Mobile Opportunistic Networks: Architectures, Protocols and Applications.* CRC Press, 2016.

[13] Nahideh Derakhshanfard, Masoud Sabaei, and Amir Masoud Rahmani. Sharing spray and wait routing algorithm in opportunistic networks. *Wireless Networks*, pages 1–12, 2015.

[14] Michael S Desta, Esa Hyytiä, Ari Keränen, Teemu Kärkkäinen, and Jörg Ott. Evaluating (Geo) content sharing with the ONE simulator. In *Proceedings of the 11th ACM international symposium on Mobility management and wireless access*, pages 37–40. ACM, 2013.

[15] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple o (log log (rank))-competitive algorithm for the matroid secretary problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1189–1201. SIAM, 2015.

[16] Thomas S Ferguson. Who solved the secretary problem? *Statistical science*, pages 282–289, 1989.

[17] PR Freeman. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.

[18] 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.

[19] Peng Jiang, John Bigham, and Eliane Bodanese. Adaptive service provisioning for emergency communications with DTN. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 2125–2130. IEEE, 2011.

[20] Denis V Lindley. Dynamic programming and decision theory. *Applied Statistics*, pages 39–51, 1961.

[21] Ramon Martí, Sergi Robles, Abraham Martín-Campillo, and J Cucurull. Providing early resource allocation during emergencies: The mobile triage tag. *Journal of Network and Computer Applications*, 32(6):1167–1182, 2009.

[22] Abraham Martín-Campillo, Jon Crowcroft, Eiko Yoneki, and Ramon Martí. Evaluating opportunistic networks in disaster scenarios. *Journal of Network and Computer Applications*, 36(2):870–880, 2013.

[23] Abraham Martín-Campillo, Carles Martínez-García, Jordi Cucurull, Ramon Martí, Sergi Robles, and Joan Borrell. Mobile agents in healthcare, a distributed intelligence approach. In *Computational Intelligence in Healthcare 4*, pages 49–80. Springer, 2010.

[24] Steven McCanne and Van Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, pages 2–2. USENIX Association, 1993.

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

[26] Stephen S Nestinger, Bo Chen, and Harry H Cheng. A mobile agent-based framework for flexible automation systems. *IEEE/Asme Transactions on Mechatronics*, 15(6):942–951, 2010.

[27] ML Nikolaev. On a generalization of the best choice problem. *Theory of Probability & Its Applications*, 22(1):187–190, 1977.

[28] Biren Patel and Vijay Chavda. Comparative Study of DTN Routing Protocols. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(5), 2015.

[29] Aniket Pramanik, Biplav Choudhury, Tameem S Choudhury, Wasim Arif, and J Mehedi. Simulative study of random waypoint mobility model for mobile ad hoc networks. In *Communication Technologies (GCCT), 2015 Global Conference on*, pages 112–116. IEEE, 2015.

[30] Adrián Sánchez-Carmona, Sergi Robles, and Carlos Borrego. Endeavouring to be in the good books. Awarding DTN network use for acknowledging the reception of bundles. *Computer Networks*, 83:149–166, 2015.

[31] Keith L Scott and Scott Burleigh. Bundle protocol specification. RFC 5050 (Experimental), November 2007.

[32] Vasco NGJ Soares, Joel JPC Rodrigues, and Farid Farahmand. GeoSpray: A geographic routing protocol for vehicular delay-tolerant networks. *Information Fusion*, 15:102–113, 2014.

[33] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

[34] Shashank Srivastava, Divya Kumar, and Shuchi Chandra. Trust analysis of execution platform for self protected mobile code. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 1904–1909. IEEE, 2015.

[35] Susan Symington. Delay-tolerant networking metadata extension block, 2011.

[36] M Tamaki. Recognizing both the maximum and the second maximum of a sequence. *Journal of Applied Probability*, pages 803–812, 1979.

[37] Sergio Martínez Tornell, Carlos Miguel Tavares Calafate, Juan-Carlos Cano, and Pietro Manzoni. DTN Protocols for Vehicular Networks: An Application Oriented Overview. *IEEE Communications Surveys and Tutorials*, 17(2):868–887, 2015.

[38] Athanasios V Vasilakos, Yan Zhang, and Thrasyvoulos Spyropoulos. *Delay tolerant networks: Protocols and applications.* CRC press, 2016.

[39] Felix Wortmann, Kristina Flüchter, et al. Internet of things. *Business & Information Systems Engineering*, 57(3):221–224, 2015.

[40] Jie Wu and Yunsheng Wang. *Opportunistic Mobile Social Networks.* CRC Press, 2014.

[41] Peng Yang and Mooi Choo Chuah. Context-aware multicast routing scheme for disruption tolerant networks. In *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 66–73. ACM, 2006.

[42] Zhong Zheng and Yijie Wang. SemanticCast: Content-Based data distribution over self-organizing semantic overlay networks. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*, pages 42–49. IEEE, 2010.