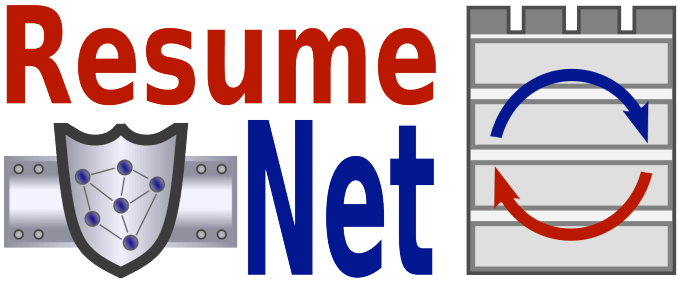

# Resilience and Survivability for future networking: framework, mechanisms, and experimental evaluation




| | |
|---|---|
| Deliverable number | WP4-D4.2b |
| Deliverable name | Deliverable on WP4 experimentation (*final*) |
| WP number | 4 |
| Delivery date | Dec. 2011 |
| Date of Preparation | 13/1/2012 |
| Editor | G. Popa (ETHZ) |
| Contributor(s) | A. Fessi (TUM), A. Fischer (UNI PASSAU), C. Lac (FT), S. Natouri (FT), G. Popa (ETHZ), Ch. Rohner (UU) |
| Internal reviewer | Prof. Bernhard Plattner |

## Summary

This document presents the the set of experiments performed in WP4 on a per-scenario basis. Each scenario tests the $D^2R^2+DR$ strategy for a number of practical situations thereby proving that it is generally applicable. The final results obtained during activities in work package 4 are presented and explained in the context of the proposed network resilience strategy. They show how various components are mapped in practice to the functional blocks of the $D^2R^2 + DR$ concept. An interesting observation is that the mapping may have a different form, depending on the scenario. For instance, the first scenario shows that in self-enforcing protocols, defense is the dominant component which fulfills the main functions, the remaining blocks having minimal functionality. For each scenario, the conclusions and lessons learnt regarding the practical use of the proposed resilience strategy are gathered in a separate section. Finally, since the opportunistic networks are a particularly challenging setting, we give a more formal account of the experiments through the four papers attached in the appendix.

## Contents

# 1 Introduction

This document presents the experimentation activities implemented in work package four since the submission of the special deliverable regarding the plan for experimentation activities. During this reporting period, the experimentation activities have been successfully finalized. Through them, the applicability of the $D^2R^2 + DR$ strategy, proposed initially by the ResumeNet project, has been thoroughly evaluated on a set of five network scenarios.

Since the $D^2R^2 + DR$ strategy is a set of guidelines and a conceptual tool, our goal was to evaluate how these concepts can be used to implement practical resilience mechanisms for realistic network scenarios. The actual defense, detection, recovery and remediation algorithms had to be designed for each scenario. Therefore, the $D^2R^2 + DR$ strategy becomes the operation template for each resilience mechanism, but the actual algorithms have to be adapted to the set of challenges that we want to address. As the outer elements (diagnosis and refinement) of the proposed resilience strategy usually involve a modification of the entire system (for instance, by a system administrator or programmer), they have not been addressed by our tests.

Each of the five experimentation scenarios is fully documented here using a common structure. Firstly, the research problem is briefly described. Note that more detailed information on this topic can be found in the special deliverable requested by the reviewers. A second section explains the experimentation methodology, the metrics that have been used and the results. Finally, each scenario provides a set of lessons learnt and conclusions that aim at showing how our resilience strategy can be used in practice.

Our tests have included two types of network scenarios. A set consists of experimental activities related to virtualization, VoIP and resilience for enterprise networks. They target concrete problems which usually appear in enterprise environments such as streaming applications ran by telecommunication operators for their clients or resilience of VoIP services. We provide a comprehensive evaluation of resilience mechanisms involving migration of virtual machines and examine the particular case of CoSIP, a mechanism proposing the use of P2P networks for resilience of voice applications. Virtualization and its applications to resilience have been studied by University of Passau and Technische Universität München. France Telecom has worked on evaluating their own implementation for network resilience with applications to enterprise networks.

A second set of use cases refers to decentralized wireless networks (wireless mesh networks and opportunistic networks). While these are not widely used in practice, there are specific cases where they are the only communication paradigm that can be used. For instance, the lack of infrastructure or intense jamming on theaters of operations may only allow the use of short-range communication; given enough mobility of hosts, this can be modeled as an opportunistic networks. Efficient communication in challenged, opportunistic networks has been studied by University of Uppsala, whereas forwarding selfishness for community-based wireless mesh networks has been explored on the TIKnet testbed at ETH Zurich.

We believe that the activities performed during the last year in work package 4 have allowed us to verify the applicability of concepts developed in work packages 2 and 3 to real situations. Our conclusions capture the fact that our general resilience strategy is applicable to a wide variety of use cases, but also that there are interactions between its components which we did not anticipate. All the details are offered by the corresponding scenarios.

# 2 Wireless Mesh Networks

## 2.1 Abstract

The issue of selfishness has become central in networking, being reflected in many game-theoretic problems: in cognitive radio, MAC layer selfishness or selfishness in resource allocation etc. In this experimentation scenario we address a specific type of selfishness (refusal to perform forwarding at network layer in order to preserve bandwidth and energy for own use), usually referred to as *forwarding selfishness* or *selfish routing*. We focus on this problem and demonstrate the applicability of the ResumeNet multi-level $D^2R^2+DR$ approach to selfishness in networks. The other three experimentation scenarios track the applicability of these concepts in other areas of networking.

The unwillingness of wireless mesh nodes to spend resources for forwarding data in a multi-hop fashion for the benefit of other hosts has been studied by us as part of work package two [PGLK10]. In WP2 our efforts have focused on the theoretical study of a reciprocation-based incentive protocol for wireless mesh networks. In WP4 we took this research one step further, by implementing the given mechanism in a typical setting (the TIKnet wireless testbed [TIK11]). By this, we provide a practical mechanism of dealing with usual situations where forwarding selfishness occurs due to the fact that hosts lack the knowledge that cooperation could be indeed achieved by reciprocation. The research question is finding out how the degree of cooperation[1] depends on the traffic matrix of the network and how our incentive mechanism performs when coupled with a standard routing algorithm operating in a realistic setting where links are affected by unpredictable errors (caused by fading, shadowing etc.).

## 2.2 Experimentation methodology, results and interpretation

The experimentation methodology is based on the selfishness avoidance algorithm presented in [PGLK10]. This technique can be applied to medium-sized community wireless mesh networks, where users are not interfering with the functions provided by the routing algorithm. Building on top of this observation, we have implemented the protocol on our existing testbed [TIK11] and quantified its efficiency (measured in terms of forwarding cooperation) under realistic conditions. One key difference between the theoretical framework [PGLK10] and the testbed implementation is the way routing is done. While in [PGLK10] the theoretical framework assumes equal links and shortest path routing, our experimentation tests the concept for a realistic network, where links are affected by multi-path effects, reflection etc. typically encountered in community network environments.

To provide a global view of the network and to enable nodes to construct a valid dependency graph, we have decided to employ OLSR as routing algorithm. The routing algorithm provides therefore global network information (including link quality as measured using ETX [DCABM03]) to all nodes in the network. Routes are therefore not anymore min-hopcount, but are chosen based on the quality of each link. Incentives depend on routes and traffic matrix, because these two elements are used to build a global dependency graph. Since it would be difficult to simulate conditions such as packet loss, error bursts and their impact on routing, the experimentation offers an assessment of the incentive algorithm's [PGLK10] suitability in realistic conditions.

[1] This concept will be defined precisely in the next sections

### 2.2.1 Building incentives on dependency graphs

We specified that we are concerned with the degree of cooperation in a given wireless mesh network, in this case, the TIKnet testbed. To this end, we adopt a metric which describes the degree of cooperation for a (topology, traffic matrix) pair as follows

$$C = \frac{N_f}{N_t}$$

where $C \in [0; 1]$, $N_t$ is the total number of data flows in the network and $N_f$ is the number of data flows that are functional. A functional flow equates to a route where all nodes cooperate, that is, they forward the information both upstream and downstream between the source and the destination of the given route. Cooperation of the relays is determined as shown in the following example.

Consider Figure 1(a), which shows a simple network topology with eight nodes, which would have to serve four flows. Each of the four flows, $k_i = \{s^i, t^i\}, (\forall) i = \overline{1,4}$ consists of a source-destination pair ($s^i$ are sources, $t^i$ destinations). A possible set of routes under shortest-path routing would be $\mathcal{S} = \{p^1, p^2, p^3, p^4\}$, where $p^1 = \{s^2, t^3\}$, $p^2 = \{t^4\}$, $p^3 = \{s^2, t^4\}$ and $p^4 = \{s^1\}$ are the routes for flows 1, 2, 3 and 4, respectively. The resulting dependency graph $G_{dep}(\mathcal{S})$ is depicted in Figure 1(b) and is computed as follows: once routes have been put in place, each route (i.e. flow) will be dependent on all the other routes having one end on this route.

The motivation comes from the fact that when a route passes through a node that acts as source or destination for another flow, the operation of the latter will depend on how the node balances its function of source/destination against its function as relay. Therefore, the dependency graph captures the implicit dependencies existing between flows in wireless mesh networks.

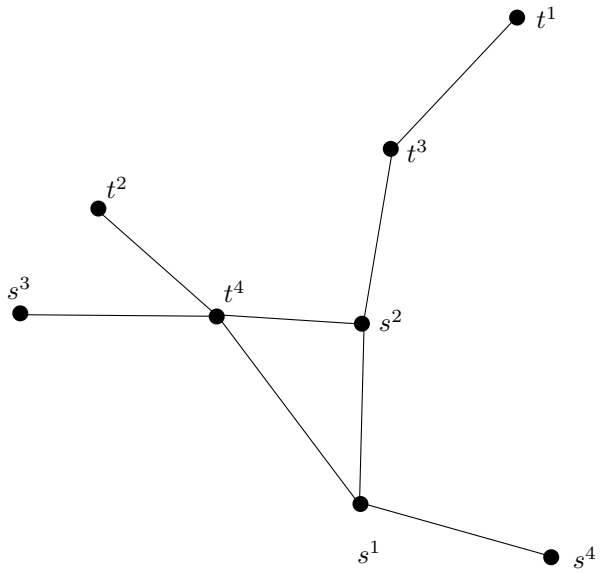
Cooperation is determined as follows:

1. strongly connected components of the dependency graph are computed; they represent flows that cooperate with each other;
2. end nodes of flows will act as relays for all other flows in the same strongly connected component of the dependency graph.

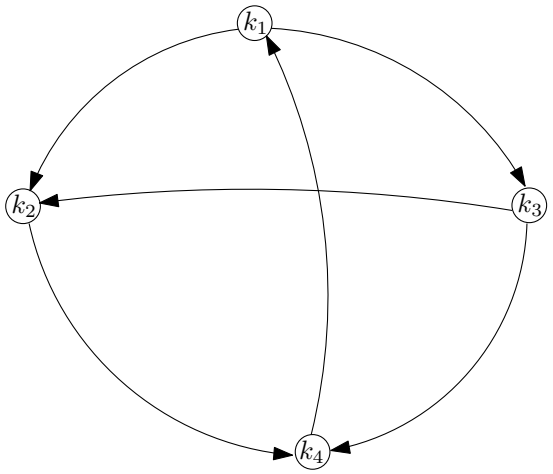
Of course, route/flow $f$ consisting only of relays which are either sources or destinations of other flows in the same strongly connected component as $f$, **is considered to be a functional flow**. This is because relays on this route are interested in providing forwarding for $f$. The cooperation degree (or ratio) is the fraction of the total number of flows that is actually functional. This is the definition that we will be using in the subsequent sections.

### 2.2.2 Experimentation methodology

In this paragraph we describe the methodology for our experiments. The proposed incentive protocol assumes link state routing (OLSR) is up and running on all hosts and that each of them knows therefore the topology of the network. In theory an advanced user could tamper with the OLSR implementation and manipulate it to lie about routes in order to obtain better results. However, in practice the type of users subscribing to such a community mesh network would most likely not have the knowledge to subvert the operation of OLSR to their own




(a) Sample topology with four flows




(b) Dependency graph resulting from the network scenario shown in Figure (a)

Figure 1: Constructing the dependency graph for a given (topology, traffic matrix, routing) tuple

advantage. Thus, each station is considered to take the information provided by the routing algorithm as correct.

Our network consists in 18 GNU/Linux hosts. Each of them is connected to the following two networks:

1. the internal wired network belonging to the TIK institute;
2. the wireless mesh network running data transfers between stations on top of the incentive protocol (10.0.0.0/24).

Wired connections are used for administrative purposes and for triggering experiments. Configuring interfaces or routing and issuing commands is done via **ssh** connections to avoid the necessity of connecting the tester's computer to the mesh network and to allow a quick

recovery in case certain hosts get disconnected from the mesh network due to propagation problems. The host **tik-wifi5** represents the central point, whose role is to set up and control the experiment. To simulate the activity of the hosts without the need to log onto each of them, we configure all the remaining hosts to trust the key belonging to **tik-wifi5** thus allowing passwordless ssh connections. Then, commands to be executed on all (or some of the stations) will be issued on **tik-wifi5** using tools such as **pdsh(1)**, which connect automatically to the specified workstations and execute as root the given commands, simultaneously on all these hosts.

*Example 1:*

```
tik-wifi5:~# pdsh -R exec -w 10.0.0.[1-8] ssh -x -l %u %h hostname
10.0.0.5: tik-wifi5
10.0.0.6: tik-wifi6
10.0.0.2: tik-wifi2
10.0.0.1: tik-wifi1
10.0.0.4: tik-wifi4
10.0.0.3: tik-wifi3
10.0.0.8: tik-wifi8
10.0.0.7: tik-wifi7
```

The wireless mesh network is used for running the experiments themselves. The steps involved in each experiment are summarized below, in the order of their execution:

1. we create at random $N_t$ (source, destination) pairs, with nodes drawn from a uniform distribution; this step is triggered by a distributed command issued through **pdsh(1)** on **tik-wifi5**;
2. each host computes the route from itself to each of the destinations it is associated with[2];
3. the routes are submitted to **tik-wifi5** which makes them available to the network;
4. based on the topology and the public routes obtained from **tik-wifi5** via the mesh network each node is now able to compute the dependency graph corresponding to the entire network[3]. The dependency graph will be the same on every host, but communication with **tik-wifi5** is necessary in order to fetch data bout the traffic matrix, which could not be obtained locally;
5. each host computes the strongly connected components of the given dependency graph;
6. hosts configure their iptables to reject all packets from or to hosts outside their strongly connected components, with the exception of management packets (including OLSR frames) which are always allowed;
7. sources probe the functionality of the route using ICMP packets; if the outcome is a 100% packet loss, then the flow is recorded as not being functional.

[2]This can be easily done by processing the routing table stored locally at the node or by performing a TCP traceroute request. Both variants have been used during experimentation

[3]Note: messages through which hosts deliver meta-information about their routes to *tik-wifi5* and by which *tik-wifi5* makes this information available to the network are not dropped by any node and are considered to be management packets. The quantity of information transported during this phase is small and nodes do not manifest themselves selfishly with respect to these packets.

Each node has three identifiers:

1. host name (of the form tik-wifiXX), associated to the wired interface and not used for experiments;
2. IP address for the WiFi card (10.0.0.XX);
3. an identifier YY such that $\{YY|YY$ – host identifier$\} = \{1, 2, 3, \ldots, N\}$, where $N$ is the total number of nodes.

Using the YY identifiers helps building the dependency graph, since these are necessarily consecutive numbers, while XX identifiers are not due to legacy issues. Hosts contain a table (one-to-one mapping) to implement the translation between the two types of identifiers.

The components running on each host are:

1. a program that computes the dependency graph; its inputs are routes and source destination pairs and outputs nodes that should be blocked by iptables;
2. random generator for (src, dst) pairs;
3. scripts to transform a node into an altruistic relay (unconditionally) or selfish host (conditionally, based on the output provided by component 1.);
4. scripts to cleanup an experiment (files containing the input for *iptables*, dependency graphs), step needed for starting a new test;

**Additional tools**

Other tools that have been used to set up the testbed are:

1. nodecontrol [TIK11]– a plugin for Webmin[4], which allows to control the WiFi card configuration on each hosts and to run tests;
2. tudp [TIK11] – custom measurement tool for UDP packets implementing ACK-less MAC-level broadcast to test connectivity.

An example, where the *nodecontrol* tool resides on **tik-wifi4** and tests connectivity between stations tik-wifi{1,2,3,4} can be seen in Figure 2.

### 2.2.3 Experimental results

The tests have consisted in placing an ever increasing number of flows in the network (see Figure 3) and recording the number of functional flows, according to the methodology detailed in the previous section. The experimental results are presented in Figure 4 and show that unsurprisingly, the expected number of functional flows increases monotonically. For lower numbers of flows, not all nodes in the network are involved as source or destination, which results in very low cooperation figures.

It is interesting to note that when the number of flows in the network is larger than double the number of nodes, then we can expect all flows to be functional. The reason is that some

[4]Webmin is a web-based platform for managing remote computers. More details can be obtained at: http://webmin.com/

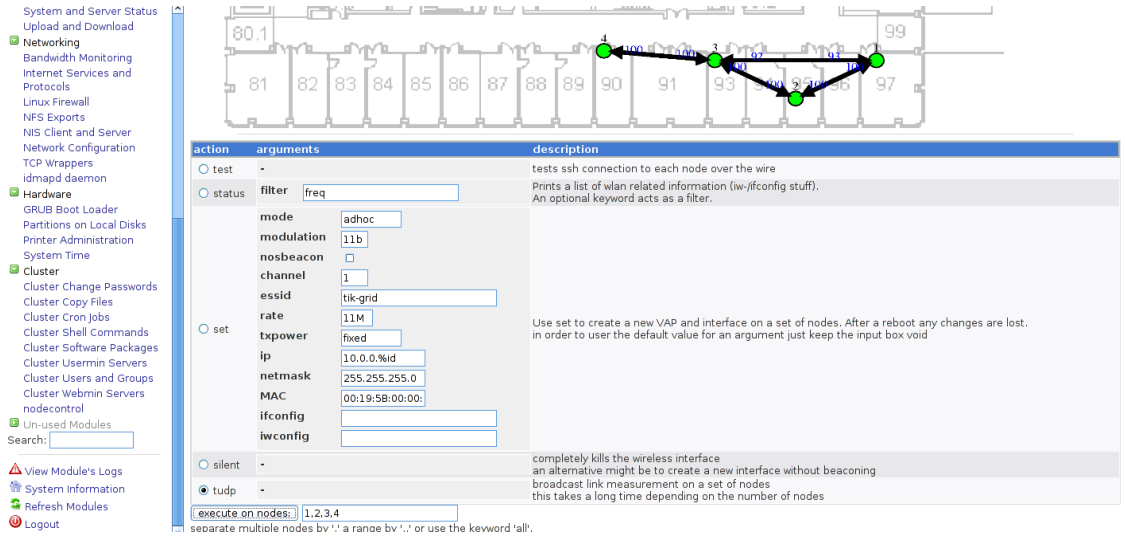



Figure 2: Screenshot – nodecontrol plugin for Webmin

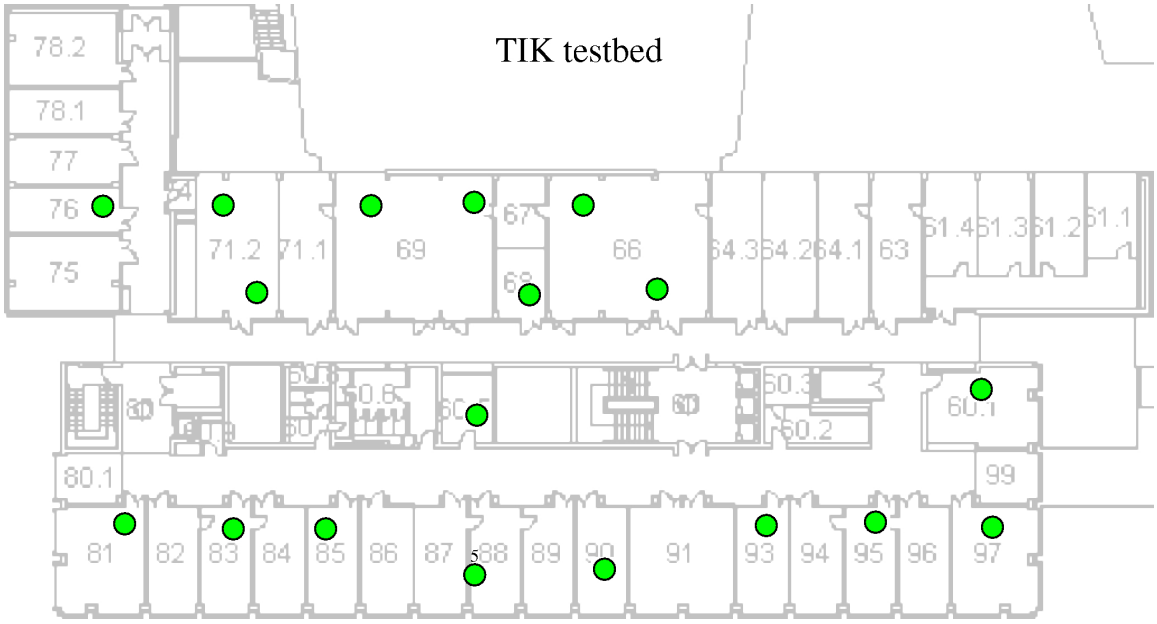



Figure 3: TikNet wireless mesh network layout

of the paths are consistently good and therefore are often chosen by the routing algorithm. Therefore, even for such a small number of flows (an average of two per node), the routes intersect each other quite a lot and therefore contribute to creating a dense dependency graph. As the dependency graph becomes denser, the strongly connected components become larger and eventually the entire graph forms a single strongly connected component. Another factor that contributes to this situation is that the concrete walls prevent propagation and therefore node degrees are relatively small (usually two or three) which produces more overlapping paths. Finally, our topology does not have isolated nodes or marginal nodes (nodes of degree 1). The results assume that all flows are started simultaneously and that all nodes are selfish. In practice, both assumptions can be relaxed, but the creation of a new flow would have to be dealt with by a new adaptive component.

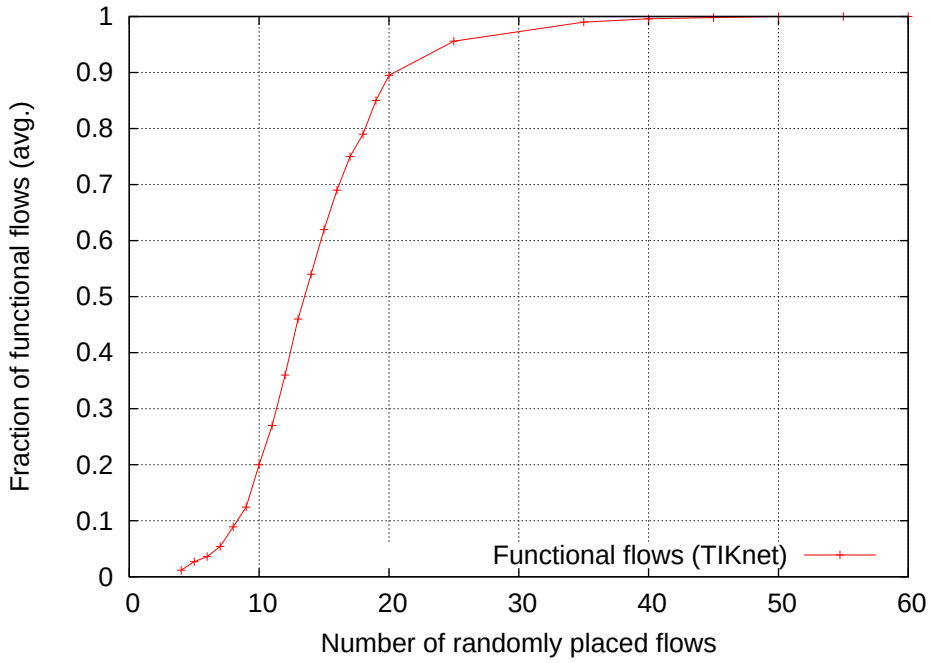



Figure 4: Average fraction of functional flows resulting from a one-shot dependency graph as function of the number of randomly placed flows

Our tests revealed that topologies where traffic is more uniform (each node is either source or destination for at least one flow) benefit from better cooperation. In fact, by definition, only nodes that expect to receive packets from the network or intend to use it for transmitting own packets should be considered as part of the network. The non-participating nodes may just as well turn off routing, since they do not intend to use the network anyway.

The results in Figure 4 generalize for other networks as well. There is indeed a dependency between the cooperation curves and the network topology. Unfortunately, the complexity of the cooperation algorithm makes the result almost impossible to compute analytically. The relation between network sparsity and cooperation figures is also difficult to assess analytically, due to the same reason.

### 2.2.4 Alternative approaches

Another approach would be to use network coding, as detailed in [KRH+06] or using random linear network coding [DMC06]. This approach has been tested partially on the TIKnet testbed [TIK11]. However, it has a number of drawbacks. The main problem arises due to the fact that the flows have to match their transmission rates and implement a strict tit-for-tat (equal quantities of data must flow on each of the routes), which results leads to much less flexibility. In the current implementation, flows are still allowed to throttle their traffic, according to the

reciprocation derived from the dependency graph. This allows one node to pay a higher price simply because it has no other cooperation opportunities, which is however a natural outcome (higher price results from unprivileged position).

## 2.3 Conclusions and lessons learnt

The results show that under reasonably practical assumptions a relatively good level of cooperation can be achieved. Calculating dependencies leads to no more and no less cooperation than that implied by the matrix of source-destination pairs. For instance, no matter which mechanism is employed, a node with only one active wireless link in the mesh will not be able to obtain the cooperation of others since it has nothing to offer in exchange. This issue could be solved by employing a payment mechanism, but in this case hosts are not anymore equal, since one has to become client and the other a provider. In a community network where all nodes are selfish by default, but also equal functionally (none of them is client/provider) and are only interested in their own traffic, such marginal nodes would be excluded. We take this as a more practical assumption, which is reflected by the operation of our incentive protocol.

Regarding the performance in terms of cooperating flows, we would like to insist that it is highly dependent on the traffic matrix. The more flows, the more likely cooperation is to be achieved. The length of these flows is a factor as well; long flows generate a lot of dependencies which may often result in better cooperation. In this report we have treated the case where the flows that are excluded from the network do not attempt to adapt their route. Such an approach would lead to significant computational overhead [PGLK10]. However, reacting to this event is possible, given that the traffic matrix does not change during route recomputation.

The results we obtained in WP2 and WP4 suggest a rather unusual mapping between the incentive algorithm and the $D^2R^2 + DR$ strategy. Incentive mechanisms are by definition not reactive. In particular, the protocol implemented here builds the dependency graph needed to protect nodes from free-riding, thereby turning them into selfish forwarders. Otherwise said, the protocol is a predominantly defensive, self-enforcing measure, as is the case of most such mechanisms. The algorithm supports the naturally occurring dependencies, which are at the basis of cooperation. The protocol proposed and tested by us cannot remediate selfishness in the network completely, but certain conditions (increasing number of flows) can bring cooperation figures very close to the maximum possible.

Usually, if one flow improves its situation (becomes functional), then a strongly connected component is likely to be broken. Therefore, remediation actions of a node can impact one or more flows and can hurt the network as a whole. This is typical for such game theoretic settings. However, if the number of follows is high, many strongly connected component merge and form with a higher probability a strongly connected dependency graph. Therefore, the actions of a player that individually adjusts its strategy are less likely to impact a network with a high flow density.

Adding a large number of dummy flows into the network can be a strategy to increase the dependencies and therefore improve globally the cooperation in the mesh network. If our objective were exclusively to improve the cooperation between nodes, then this would be a valid remediation action in the context of the $D^2R^2 + DR$ resilience strategy. A large number of active flows would hurt the real network performance and therefore cancels out the cooperation gain. A more complex remediation strategy would consist of a joint optimization of cooperation and throughput. In practice this translate to complex MIP optimizations, which are very difficult to solve even for a small number of nodes.

In conclusion, we identified a scenario (selfish forwarding) where network recovery and remediation are merged, and where defense plays a key role and is self-enforcing. Moreover, recovery and remediation have a definition very much different from the classical scenario. Helping some flows may easily hurt others and therefore recovery and remediation have to find a balance. Some topologies/traffic matrices may not even allow a good cooperation, leading to a situation which simply cannot be remedied. We prove the applicability of the $D^2R^2 + DR$ strategy by pointing out that network performance (cooperation in this case) has an upper bound deriving from the dependency graph, which is a function of the topology and traffic matrix.

# 3 Opportunistic Networks

## 3.1 Abstract

The store-carry-forward transport in opportunistic networks offers an inherent defense mechanism against the challenge of episodic connectivity. The connectivity itself offers diversity, while replication of the data introduces both spatial and temporal redundancy. Our experimentation is on different levels. First we apply the $D^2R^2 + DR$ strategy to opportunistic networks that illustrates the design space for choosing replication parameters and highlight the trade-off between resilience and resource utilization. We propose and evaluate different strategies in resource management and data replication for data-centric opportunistic networks and find that keeping buffers fresh and considering the order in which data is exchanged benefit the performance of the system. We also show that decisions based on local information is efficient and even gives a performance advantage in the short term compared with decisions based on global information. Finally, we evaluate the resilience of opportunistic networks with respect to different challenges using the metric framework for a variety of topologies and routing algorithms.

As an effect of our work, we designed an efficient trace-based performance analysis method that reduces the experimentation run-time in the order of magnitudes compared to event-driven simulations.

The results are presented in five publications (Paper A to E) that are enclosed in the appendix. In the following sections we set the work into context and summarize the results.

## 3.2 Resilience in the case of opportunistic networks

In opportunistic networks, typically mobile nodes store, carry, and forward messages when they encounter other nodes, using short-range communication. A store-carry-forward (SCF) transport service allows a flow of data in the network despite the absence of end-to-end paths. Data instead travels over *space-time paths*, comprised of sets of links that become available in different time instants in the network. Node mobility is thus important for data dissemination; it creates contact opportunities between different nodes and allows nodes to physically transport data to areas where no connectivity might be available.

Based on the ResumeNet resilience strategy, we illustrate how opportunistic networks can be enhanced to cope with challenges. Our overall strategy is depicted in Figure 5. The SCF transport service has inherent *defense* against the challenge of episodic connectivity. The mobility of nodes gives rise to a diversity of space-time paths. Depending on how aggressive a data forwarding scheme is, the network can exploit all or part of it, at the expense of resource

consumption. This same diversity can be the counter-measure against misbehaving nodes, whether selfish or malicious, which fail to contribute to the transport service.

We discuss the application of the $D^2R^2 + DR$ strategy in detail on the example of selfish node behavior in Paper A. In the enclosed experimentation we show that remediation indeed can compensate for selfish node behavior, but at the price of a high overhead. Applying resource management as a refinement mechanism allows us to mitigate for this overhead, still maintaining good data delivery.

**Paper A** *On Realising a Strategy for Resilience in Opportunistic Networks*, Marcus Schöller, Paul Smith, Christian Rohner, Merkouris Karaliopoulos, Abdul Jabbar, James P.G. Sterbenz, and David Hutchison, Future Network and MobileSummit 2010, Florence, 2010

*Experimentation*: The experimental results are obtained with the Haggle experimentation architecture running on (virtual) nodes with controlled connectivity. Connectivity between two nodes follows a two state Markov model. This topology avoids large connected clusters but still gives enough contact opportunities to exchange data. Our emulator offers the possibility of dynamically changing the forwarding strategy and resource management policies, for example to age out data carried for a long time. We use the two-hop forwarding scheme with no data aging as the default configuration (as normal operation), for remediation we change to epidemic forwarding, and finally activate data aging to demonstrate refinement. Delay and resource usage are obtained through analysis of the log files.

*Conclusion*: In this paper, we have argued that resilience is a necessary building block for any future network. Despite the fact that many resilience mechanisms have been added to networks, especially to the Internet, a systematic approach to resilience has not so far been developed in order to increase its availability and survivability. We have introduced a general strategy that aims to embed resilience systematically into networked systems. We have applied our strategy to an opportunistic networking scenario, showing some of our early results and how this strategy can enhance the network over time.

## 3.3 Experimentation methodology

The experimental research within the context of ResumeNet is two fold: First, we explore the *resilience of opportunistic networks* in general. We try to answer the question of how resilient is the SCF transport mechanism with respect to different challenges, and how the choice of replication and resource management does impact the resilience. We use the metric framework to do that analysis. Second, we investigate the *strategy selection* of defense/remediation mechanisms to find a balance between performance and resource use.

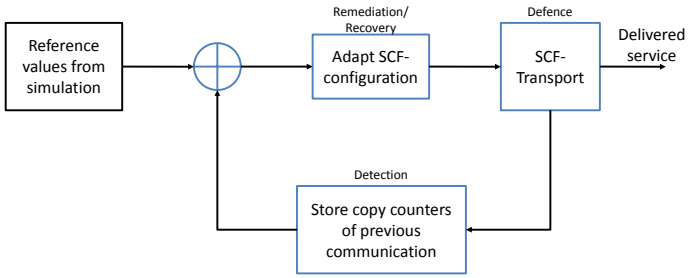



Figure 5: Control loop steering the transport protocol

Our studies on the resilience of opportunistic networks is performed with event-driven simulation, while concrete remediation and defense mechanisms are implemented in the Haggle architecture [NGR09] and studied by emulation experiments performed on the in-house Haggle testbed [BGR11]:

- The ONE simulator [KOK09] has been written explicitly for the simulation of opportunistic protocols and is frequently used by the DTN community. In our experiments we used the option to feed connectivity traces rather than using the provided mobility models to also allow studies on established real-world connectivity traces.
- The Haggle testbed is a trace-driven emulation testbed, where contact traces decide the connectivity between Virtual Machine (VM) nodes. The testbed allows us evaluate different protocols and strategies under the same network conditions.

As an effect of our work, we also designed an efficient trace-based performance analysis method that reduces the experimentation run-time in the order of magnitudes compared to event-driven simulations. This analysis method is presented in Paper B.

**Paper B** *Trace-based performance analysis of opportunistic forwarding under imperfect cooperation conditions*, Merkouris Karaliopoulos and Christian Rohner, IEEE INFOCOM mini-conference 2012, Orlando, 2012

*Description*: We propose a systematic method for efficiently approximating the performance of opportunistic forwarding protocols over real traces of mobile node encounters (contacts). At the core of our method, which bypasses discrete event simulations, lie trace-inflation and graph-expansion techniques. Through their combined use, space-time graph constructs are derived out of the contact sequences for each message and forwarding scheme. Standard shortest path algorithms can then be efficiently applied over these small sparse graphs to extract the space-time paths messages follow under the control of different forwarding schemes.

The method can compute the number of delivered messages, message delivery delay, and the shortest path hop count. It is approximative in the sense that it cannot account for resource contention within the network. Nevertheless, the match with the results of the most popular simulator for opportunistic networking is excellent and is achieved at computation run times up to three orders of size smaller than the simulator. Skipping the trace inflation process does help the run times scale better for big traces; it is then up to the users of the method to opt for this at the expense of some accuracy penalty, which was found small in our experiments. The computational savings pertain across different evaluation scenarios featuring perfect node cooperation and different expressions of selfish node behavior thus rendering the method a promising tool for the performance evaluation of opportunistic networking protocols.

## 3.4 The impact of challenges

The fact that the SCF transport does provide an inherent defense mechanism for intermittent connectivity raises the question if (and how well) it also defends against other types of challenges. Challenges that affect the usage of a node contact to transfer data between nodes due to limited abilities (e.g., interference, full buffers, low battery), node failure, active attacks by third parties, or because of selfish node behavior. We abstract from the concrete challenge by

selectively removing *node contacts* from contact traces, or excluding *nodes* from participating in the data forwarding process. We do the selection both randomly and informed:

- Removing node contacts: we remove random or spatially correlated node contacts. The spatial correlation is implemented by considering a jammer within which range we remove node contacts.
- Selfish node behavior: we exclude random or strategically selected nodes from participating in the data forwarding process. The strategic selection of nodes takes node centrality and betweenness centrality into account, thus excluding nodes that are likely to contribute most to the forwarding process under normal operation.

We use the metric framework to study the effect of a challenge. The challenge degree is varied by the number of node contacts and selfish nodes that are excluded. The performance metric used include data delivery delay and data delivery ratio. The analysis is presented in Paper C.

**Paper C** *Resilience Metric in the case of Opportunistic Networks*, Christian Rohner, Fredrik Bjurefors, Maria Mehrvapar, and Paul Smith, Technical Report

*Experimentation*: The resilience of opportunistic networks is studied by using the metric framework to derive metric envelopes on delay and the delivery ratio of messages inserted at random into the network. We gradually increase the challenges on the network by modifying the mobility traces (i.e., removing node contacts) or node configuration (i.e., selfish behavior) of the simulator and the Space-Time-Graph methodology. Resilience is studied for different network topologies using different mobility traces, and for different routing protocols.

*Conclusion*: We find that the diversity in space-time paths give a high degree of resilience to contact and node failures, offering alternative paths to deliver data. Although impact of a challenge on an individual message can be significant if a contact opportunity is missed, we find that the overall performance decrease is typically graceful until a significant amount of the node contacts or nodes are affected, depending on the diversity offered by the topology.

The behavior is similar for most of the studied forwarding protocols, with the exception of the Spray and Wait routing protocol which surprisingly even showed cases where the data delivery delay improved under challenges affecting node contacts.

We show, on the example of node selfishness, that selective challenges affecting strategic nodes and contacts have a higher impact on the system performance but, depending on the mobility trace, increases the delay with less than 20% compared to random selection.

## 3.5 Defense and remediation

The SCF transport is not only as a way to defend for intermittent node contacts, it also is a means to make use of the node contacts to disseminate data. The decisions what data to replicate and forward, and what data to drop when buffers get full have a significant impact on the ability to disseminate data and the resource usage of the system. These two aspects are commonly referred to as forwarding and resource management, respectively.

The trade-off between the ability to disseminate data and resource usage is an important part of the remediation process and is reflected in many forwarding protocols. Some limit the

epidemic spread by reducing replication [FyLBS+06], [GT02], [SPR05], others make use of contact history or social relations between nodes to identify good forwarders [LDS03], [BCIP07], [HCY08]. In our work we consider two aspects of defense and remediation that address the resource usage independent of underlying forwarding protocols, namely resource/congestion management with limited buffers (what to drop if buffers get full), and the ordering of the data transfer during time-limited node contacts (what to send first). In Paper D, we are the first to define congestion in opportunistic networks and show that the freshness of the data in a limited buffer is essential for the performance of opportunistic networks. Paper E suggests that nodes actually benefit in the short-term from ordering data transfers based on local information only.

**Paper D** *Congestion Avoidance in a Data-Centric Opportunistic Network*, Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli, ACM SIGCOMM Workshop on Information-Centric Networking (ICN2011), Toronto, 2011

*Experimentation*: Five buffer management strategies taking node interest and forwarding statistics into account were implemented in Haggle and evaluated on the Haggle testbed. Four scenarios are used in the evaluation. In each scenario, five communities are connected by different types of hub and bridge nodes to study the buffer management strategies under different forms of bottlenecks. The mobility, community graphs, and contact times are emulated and are filtered according to scenarios and connectivity traces. The user data production and consumption for each node is identical for each scenario in order to be able to compare strategies under the same conditions. We analyze delivery ratio, dissemination speed, and overhead.

*Conclusion*: We have evaluated dropping strategies at congested nodes for data-centric opportunistic networks in different community scenarios. In such networks, congestion avoidance, buffer handling, and choice of data object to drop must be based on local information. Nodes do not have complete information of the interests of other nodes in the network since Node Descriptions will only spread between nodes that share interests. From the experimental evaluation, we conclude that the strategy which drops the data object with the highest number of replications made at the local node performs overall best with respect to delivery ratio, delay, and overhead at a congested node, i.e., when there is not enough buffer space for relaying objects. Furthermore, using local replication information when dropping gives a higher delivery rate compared to using local interest information. The strategy to randomly select a data object to drop comes surprisingly close to the best cases in our measurements. This shows that already with a small relay buffer, performance gains can be achieved with a dropping strategy where data objects often are exchanged.

**Paper E** *Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks*, Christian Rohner, Fredrik Bjurefors, Per Gunningberg, Liam McNamara, Erik Nordström, ACM Workshop on Mobile Opportunistic Networks (MobiOpp2012), Zürich, 2012

*Experimentation*: Five strategies to order data during a node contact were implemented in Haggle and evaluated on the Haggle testbed. We use three traces with different properties to minimise the bias of our results due to the configuration of a particular trace, and understand transfer ordering across a range of network densities and churn. We distribute interests and data according to a well-known model of data and interests on the web (i.e., Zipf distribution). The normalized Discounted Cumulative Gain, a relevance metric from the area of information retrieval, allows us to evaluate the relevance of the

data delivered to nodes in our experiments. We analyze delivery ratio, received data relevance, and transfer efficiency.

*Conclusion*: This paper proposed that the ordering of data items in transfer limited data-centric opportunistic networks has a large impact on the relevance of received data. We experimentally compared five strategies that select and order data to be transferred and evaluated their ability to deliver the most relevant data, measured using the normalised Discounted Cumulative Gain (nDCG). The trace-driven experiments on an emulation testbed included contact traces with a range of densities and mixing.

We saw that selection and ordering of transferred items significantly affects the achieved delivery ratio and its rate of increase, respectively. If nodes desire highly relevant data, careful ordering of limited transfers can satisfy these desires quicker than order agnostic approaches. With local strategies, if it is unlikely that neighbours are interested in the same content, segregation can occur where nodes are unable to disseminate items through the network. Interestingly, considering the relevance of data provides a way to overcome this segregation. In the case of densely connected networks, just local decisions can achieve system properties. However, in particularly sparse networks we need to explicitly consider non-local information to successfully promote system concerns.

We saw how selecting items to transfer using global information eventually outperforms local approaches. However, this does not mean that changing strategies in the experiment achieves the best of both approaches, as the early transfers dictate later performance.

## 3.6 Detection

Challenge detection in opportunistic networks is not straightforward because normal operation is not defined and because end-to-end connectivity for verification are not available. Our most promising approach was to explore the possibilities and limits in a simplified scenario, such as two-hop forwarding where the forwarding path is simple enough to be verified. To this purpose, verifiable service agreements could be used where nodes commit to carry messages for a certain period of time, and selfish node behavior could be detected by verifying if messages were passed on during contacts before the agreed time expired when source and destination nodes meet later on. While carrying messages for a longer time implies that more node contacts can be used to exchange the message, which results in a higher delivery ratio and a lower delay, it also increases the number of messages a node is expected to carry and is thus likely to fill up buffers – making it difficult for the detection mechanism to distinguish selfish node behavior from node congestion. A time short enough to avoid congestion but long enough to not affect performance too much would thus be desirable.

We explored the design space for such verifiable service agreements by studying the impact of the duration to carry messages (timeout) on the end-to-end delivery delay and delivery ratio. Figure 6 summarizes the results for epidemic, 2-hop, and source spray and wait (ssaw) forwarding for the Infocom05 trace [SGC+09], indicating a significant penalty on the system performance. Figure 6 also shows the average hop count and number of copies per message. A high number of hops indicates that messages are passed from node to node to keep them "alive". Epidemic forwarding shows a peak at a timeout around 250s; higher timeouts increase the likelihood to keep a message until an optimal node contact arises, smaller timeouts letting the message to expire on many nodes before good node contacts arise. The number of copies per message suggests how many times a message is put back into the system after it has

expired. The numbers for 2-hop and epidemic forwarding increase in orders of magnitude for decreasing timeout making the system inefficient. The results suggest that meaningful service agreements would lead to a significant performance degradation and increased overhead.

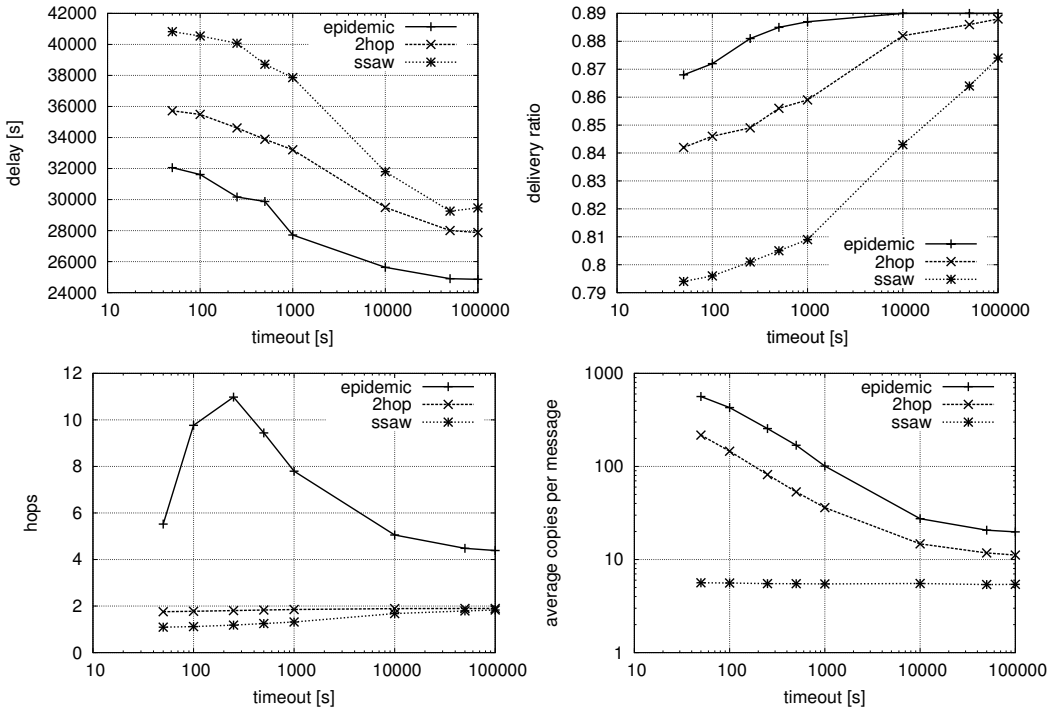



Figure 6: Impact of buffer-timeout on (a) delay (b) delivery ratio (c) hop count (d) number of copies.

## 3.7 Conclusions and lessons learnt

The store-carry-forward functionality provides a robust defense mechanism not only for intermittent connectivity but also for challenges like jammers, contact losses, or selfish node behavior. The observed performance degradation is typically graceful due to the diversity in space-time paths that offer alternatives when contacts or nodes fail. Surprisingly, spray and wait forwarding even showed a performance improvement when dealing with selfish nodes, suggesting that the first contacts not always need to be the most promising to carry on data.

Dealing with resource limitations is crucial in opportunistic networks and an important part of the remediation process. We show that the freshness of data in limited buffers is important for optimizing data delivery. Also, we point out the importance of data selection and transfer ordering under time-limited contacts. Our results suggest that nodes benefit in the short-term from ordered data transfers based on local information, while selection based on global information benefit long-term system performance.

We applied our general resilience strategy $D^2R^2 + DR$ to an opportunistic networking scenario, showing how it can be used to address the challenge of selfish nodes. We found detection to be the most challenging part because the typically stochastic characteristics of opportunistic networks make a reliable and timely challenge detection difficult. Also, it is not clear what the definition of normal operation should be. We therefore decided to focus on individual aspects of the resilience strategy rather than an implementation of the entire control

loop.

We found that remediation and recovery serve the same purpose, namely adapting the SCF configuration keeping performance and resource use in balance. They might be considered equivalent from a control theory point of view.

The variance in delivery delay for different data items is significant. Space-time diversity at the time when data is created is crucial and a short variation in timing can make a siginificant difference in delay. The metric framework and metric envelopes therefore allow a much more adequate interpretation of the results than average-only results. However, as the extreme values are wide spread with both immediate delivery and heavy tails, we found the use of percentiles as upper- and lower-bounds on the envelopes more meaningful than min/max envelopes.

# 4 P2P Signalling

## 4.1 Abstract

In this section, we summarize our experiments and findings in the area of P2P signaling. The case study described here deals particularly with P2P signaling for Voice over IP (VoIP) as an example application.

## 4.2 Methodology: Cooperative SIP (CoSIP)

In this section, we present the CoSIP concept as well as example application scenarios.

### 4.2.1 Concept

Unlike the P2PSIP approach discussed in the IETF, we follow a supervised P2P network approach, combining the advantages of P2P and SIP client/server architectures in order to provide better reliability, security and performance. The SIP server cooperates with the SIP UAs to manage user registrations and session establishments. The current locations of UAs are stored at the server and are additionally propagated in the P2P network (Figure 7). Lookup requests are likewise resolved by the SIP server and by the P2P network concurrently (Figure 8). As a choice for the P2P network, we use a DHT in order to achieve efficient routing and avoid flooding requests. Legacy SIP UAs that do not support CoSIP can be connected to the DHT via an adapter that we call "CoSIP proxy"'. The same holds for weak devices with less CPU or bandwidth. They are represented in the P2P network by a CoSIP proxy running on a more powerful machine. In both cases, a CoSIP proxy processes user registrations as well as session establishments for the UAs behind it.

Figure 7 and Figure 8 show the basic functionality of CoSIP. When Alice's UA registers to the SIP registrar, it also store her contact data in the DHT:

```
STORE(H(alice@example.org), alice_IP:alice_port)
```

This data will be propagated in the DHT and can be used afterwards to resolve the SIP URI of Alice to her current location. By the time another peer, let's say Bob, needs to initiate a session with Alice (Figure 8), Bob sends an `INVITE` message towards the SIP server. Additionally, Bob computes the hash value of Alice's URI and locate Alice's contact data in the DHT:

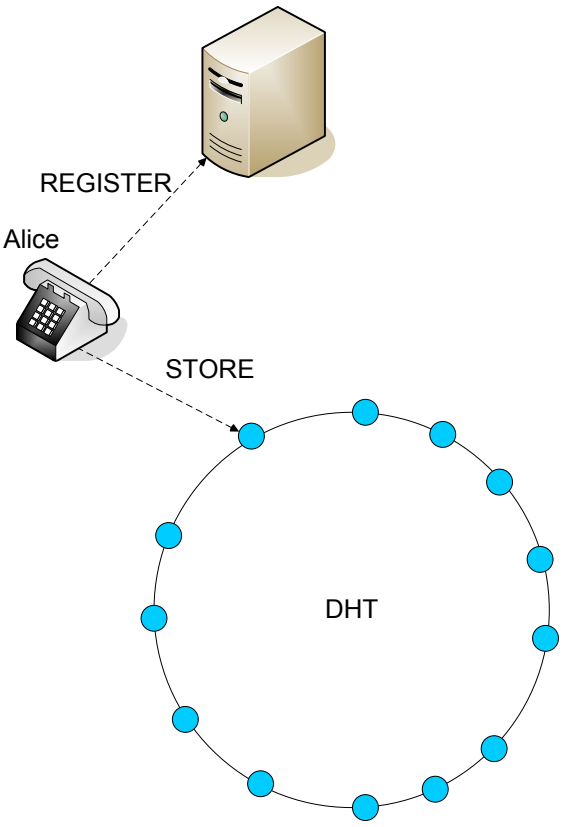



Figure 7: Registration of a SIP UA with CoSIP.

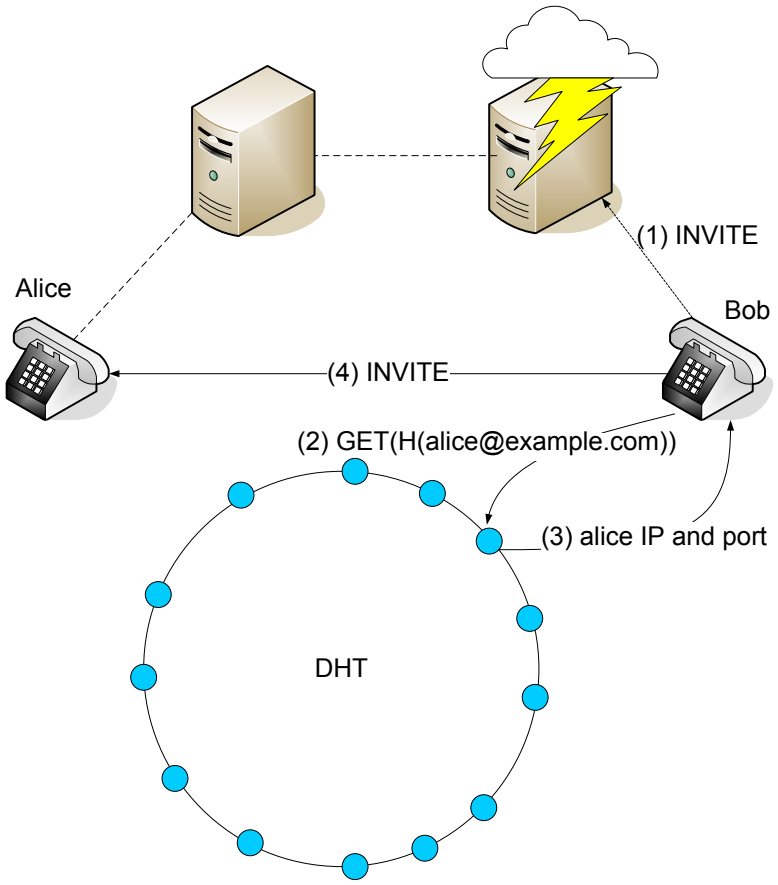



Figure 8: CoSIP session establishment in case of a server failure.

```
(alice_IP:alice_port) = GET(H(alice@example.org))
```

The `GET` request is propagated in the DHT until one of the replica nodes storing the data with the key `alice@example.org` responds.

A response from the DHT may take longer depending on the routing algorithm used for the DHT, the number of peers and the stability of the DHT. However, in case the server is unreachable due to a network or service failure, or the server is undergoing an attack or an overload situation, the response from the DHT can be very useful. When Bob receives the required information to contact Alice from the DHT, he sends an `INVITE` message directly towards Alice and the session establishment can be completed. In other words, in case the server is overloaded or unreachable, the DHT serves as backup. This provides a significant improvement to the reliability of the SIP service compared to traditional SIP. On the other side, CoSIP provides improved security compared to P2PSIP.

Concluding this section, centralized SIP infrastructures have a lack of reliability and are vulnerable to DoS attacks; P2PSIP networks are hard to secure and are vulnerable to a number of attacks such as Sybil attacks, eclipse attacks, partition attacks, or SPIT. Therefore, CoSIP is intended to fill the gap between these two solutions by combining them in order to benefit from their respective advantages.

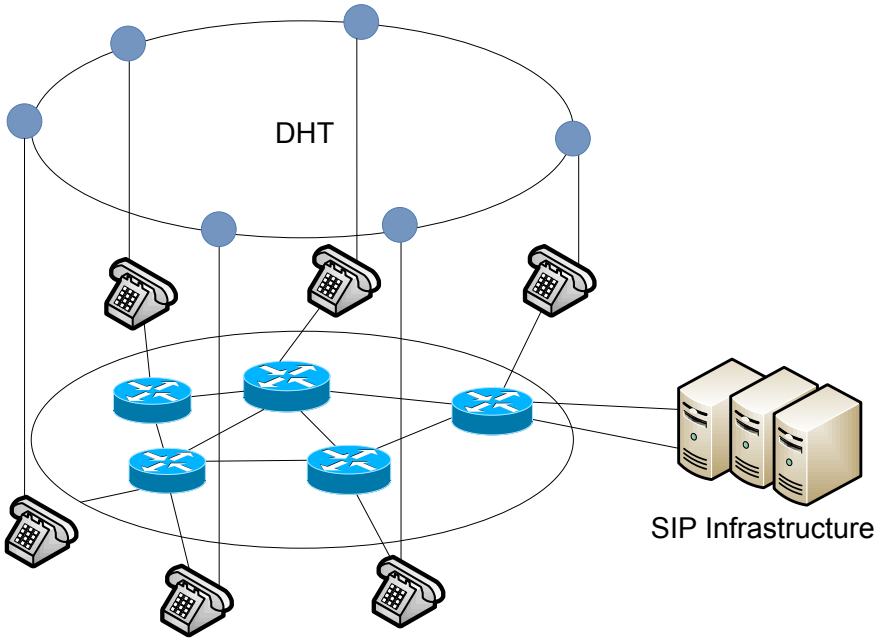



Figure 9: CoSIP operation in an enterprise network.

### 4.2.2 Application Scenarios

An application scenario of CoSIP is sketched in Figure 9. A large-scale VoIP network, e.g., an enterprise or university SIP network. SIP UAs communicate with each other during normal operation and set up a DHT. Server downtimes due to failures or maintenance can be bridged by CoSIP.

Figure 10 shows another application scenario of CoSIP. Small Office and Home Networks (SOHO) are connected to an Internet and VoIP provider via DSL routers. Each DSL router contains a CoSIP proxy which acts as an outbound proxy for end devices in its SOHO and implements the additional CoSIP functionality. The CoSIP proxies communicate with each other and organize themselves in a DHT. In the regular case, the SIP infrastructure of the VoIP provider is used to establish sessions between end devices in different SOHOs. In case the SIP infrastructure is temporarily unavailable, the DHT acts as backup and end devices can still establish phone calls.

In both scenarios described above, CoSIP provides a low-cost solution for significantly improving the reliability of the VoIP service. It is a proactive solution which does not require any challenge detection mechanisms or manual configuration when a failure at the infrastructure occurs.

### 4.2.3 CoSIP Reliability and Security Evaluation

Details about the reliability of storage and routing in P2P networks can be found in [Fes10]. In the experiments described in this section, P2P networks are used to enable application layer session setup by storing and locating contact data (IP and port) of applications or services in the P2P network.

Given the security issues incurring in a pure P2P solution, the solution that we use, CoSIP,

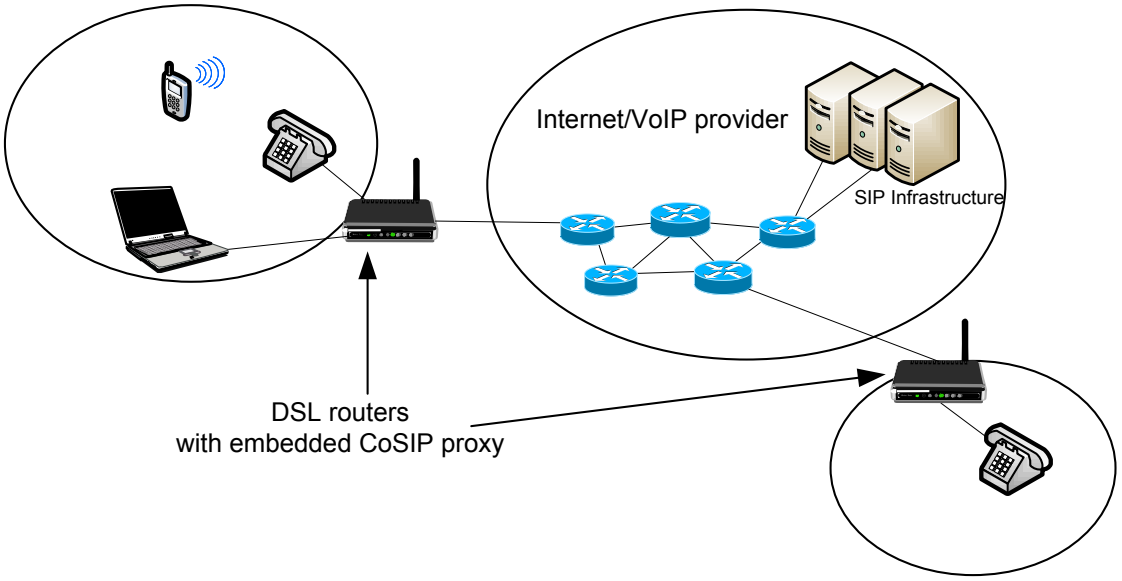



Figure 10: CoSIP operation in an Internet/VoIP provider network.

is a hybrid approach where contact data is stored in parallel at a server and in the P2P network as well. The server plays a critical role in terms of providing verifiable identities when a new peer joins the network. Under normal operation, the signaling is performed via the server. In case of service failure, the session setup can be performed via the P2P network. However, this is a remediation mechanism only. When the server is back to operation, e.g., after a successful VM migration as described in Section 5, signaling should be performed via the server again. More details on the security issues in P2P networks and how they can be addressed with the hybrid approach CoSIP can be found in [Fes10].

### 4.2.4 Relationship to the $D^2R^2 + DR$ Strategy

We briefly remind the reader how the $D^2R^2+DR$ resilience strategy is valid when using CoSIP, particularly in the context of our experiments.

**Defence** Defence mechanisms reduce the probability that a failure occurs on one hand, and reduce the impact in case of a failure on the other hand. CoSIP provides two defence mechanisms:

- The location of SIP User Agents (UAs) is stored at the SIP server as well as the P2P network, thus providing redundancy.
- In an enhanced version of our experiments, the SIP server is hosted within a virtual machine. This provides the ability to react fast to challenges. For example, upon overload, the VM can be duplicated. New duplicates of the VM can then be started with a SIP server running in it. Another option is to migrate the VM during run-time to hardware with higher capacity.

The latter case is a integration between our CoSIP approach and experiments with virtualization. More details about the virtualization part can be found in Section 5.

**Challenge Emulation** The goal is to emulate a challenge which makes the server unavailable for signaling. This can be, e.g.,

- An emulated network failure which renders the SIP server unreachable for the SIP UAs.
- A local challenge at the server, i.e., software or hardware failure, or misconfiguration, which renders the server unreachable for the SIP UAs.

We choose the former and more simple option. It can be easily implemented, e.g., either by adding a firewall rule at the host where the SIP server is running, or by simply unplugging the Ethernet cable from the host.

**Detection** The detection can take place at the different parts of the network.

- SIP UAs can detect that the SIP server is not reachable anymore, since it does not respond to their requests.
- Monitoring probes are distributed along the network and can detect that the server is not reacting.
- Using challenge detection mechanisms, an anomaly in the number of requests or responses sent to and from the server is detected.

**Remediation** Like the detection mechanisms, remediation can take place at different parts of the network.

- The SIP UAs switch to P2P mode and can get the required information for the signaling (IP and port) from the DHT.
- As for the server side, depending on the reason for the server unavailability, a possible remediation mechanism is starting a new server, either in the same location or in another network. Or migrating the server if possible. If the server is hosted within a virtual machine, live migration of the VM allows for transferring the hot state of the server. And thus in the ideal case, it should allow for the clients to continue their sessions without interruption. A discussion on the different remediation options using virtualization is in the attached paper [FFCdM11].

**Recovery** After receiving the information about the new location of the SIP server (potentially another server) the SIP UAs can switch back to normal operation and perform their signaling using the server. One of the reasoning behind going back up to server signaling is that P2P signaling is only a remediation mechanism which does not provide the same Quality of Experience (QoE) to the users. For example, the detection of Spam over IP Telephony (SPIT) is a lot more efficient at the server side.

## 4.3 Experimentation and Results

Experiments with CoSIP which we performed can be classified

- On one hand on the fact whether they were performed on local testbeds or on a distributed testbed. In the case of our experiments with CoSIP, this was PlanetLab.

- On the other hand, whether they were integrated with our virtualization experiments.

Functional and performance tests are reported below. For the integration with virtualization, the reader is referred to [FFCdM11].

We implemented CoSIP as a local SIP proxy that processes the SIP signaling of one or more SIP UAs. Implementations of the SIP UA do not need to be aware of CoSIP. The SIP UA just needs to be configured with the CoSIP Proxy as an outbound proxy. We performed extensive experiments on local testbeds as well as on PlanetLab to validate the functionality of CoSIP. The intention behind PlanetLab was having an emulated "real life-like" network. However, one of the most remarkable properties that we noticed when experimenting with PlanetLab was the high variance in the round trip time (RTT) due to the different locations of the nodes and the different load on the respective machines. An obvious limitation of PlanetLab is that we can not run a realistic VoIP network of an Internet/VoIP provider given the limited number of PlanetLab nodes.

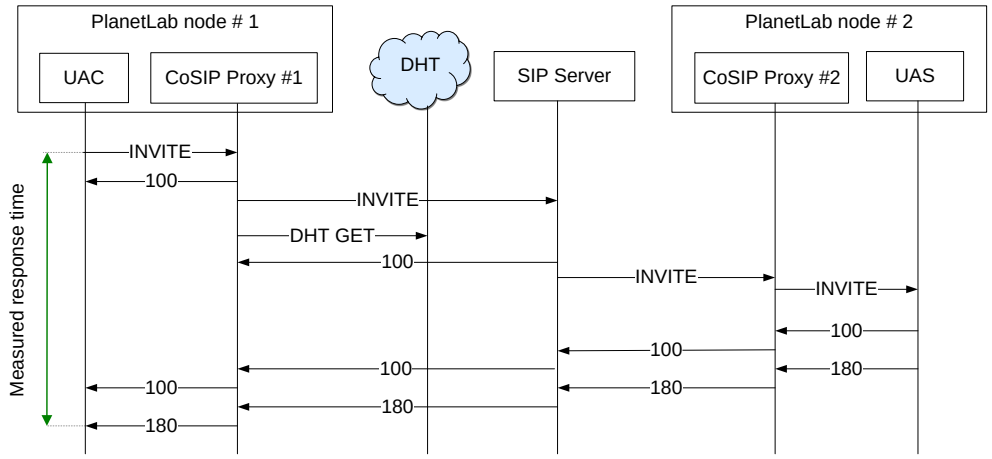
**Testbed Setup** The main motivation behind our tests was to show a survivability use-case of the SIP service, i.e., if CoSIP is used, SIP UAs are able to establish a session even if the server is unreachable. Additionally, we measured the time required to establish a session in two cases: first, under normal conditions with server-based signalling, and second, with an emulated server failure and DHT-based signaling. We compared the time for the session establishment in both cases. We consider the time required for the session establishment as the length of the time interval between sending an `INVITE` message by a User Agent Client (UAC), and receiving a `180 Ringing` message from the User Agent Server (UAS)[5].

We use the open source SIP stack PJSIP [Ope10] to generate SIP traffic. We use the PJSIP API to perform UA registration to the SIP server and initiate VoIP sessions randomly. We logs timestamps when a message is sent or received at the CoSIP proxy. We run a network of 430 PlanetLab nodes with a SIP UA and a CoSIP proxy on each PlanetLab node. Given the relatively small size of the network we performed different experiments with different size of the Kademlia $k$-buckets, particularly $k = 2, 4, 8$. The SIP Server (SIP Express Router) and a Kademlia node for bootstrapping the DHT nodes were installed on a server running in our department at TU Munich.

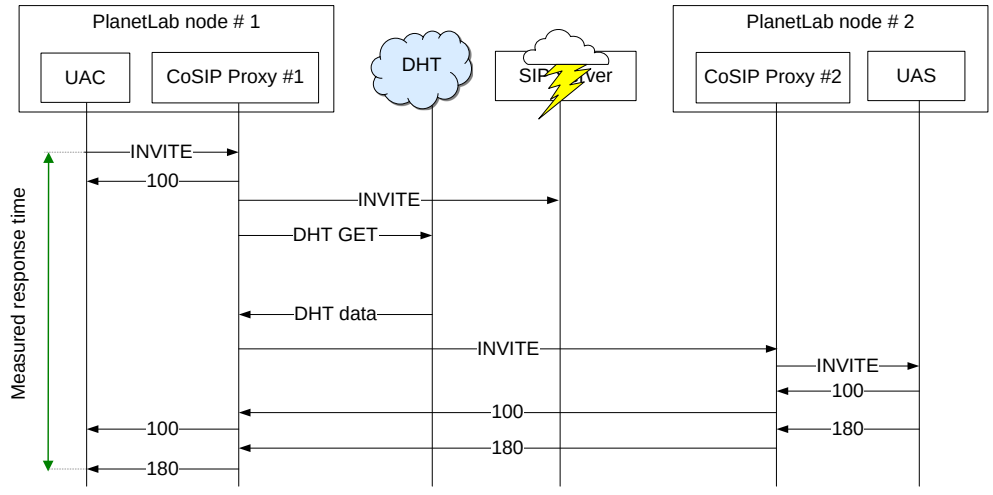
**Functional Tests** Figure 11 shows the message flow for session establishment with a running SIP server and when the server is unreachable respectively. In both cases, message flows were collected based on logs from the CoSIP proxies. The message flow on Figure 11(b) shows that the SIP network can survive and that SIP UAs are able to establish a VoIP session even when the server is down.

**Performance Tests** After a test run for 5 hours, 5,380 phone calls were emulated by the UAs on PlanetLab. All logs are exported by the CoSIP proxies periodically to a central server which collects the call records in a SQL database. Figure 12 shows the time required for session establishment with the server vs. DHT. with different $k$-bucket size $k = 2, 4, 8$. Each boxplot shows the 10%, 25%, 50%, 75% and 90% percentile. The median is approximately 0.543, 0.306 and 0.191 for the DHT with $k = 2, 4, 8$ respectively and 0.186 for the server The

[5]Waiting until a `200 OK` is received would not be appropriate as a measure, since it depends on how long the user needs to pick up the handset. Therefore, the `180 Ringing` message seems to be more appropriate as a sign that a session between the UAC and the UAS has successfully been established.




(a) SIP server up




(b) SIP server down

Figure 11: Message flow for session establishment

difference between the median values for $k = 8$ and for the server case is quite small (less than $1ms$) given that in the majority of the cases the DHT lookup can be performed within one hop.

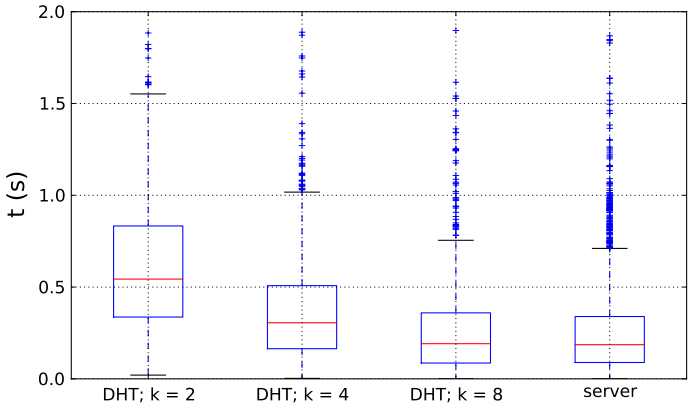



Figure 12: Time required for session establishment with server vs. DHT with different Kademlia $k$-bucket size.

### 4.4 Conclusions and Lessons Learned

In this section, we summarized our experiments and results about P2P signaling, notably signaling for VoIP. CoSIP is our approach to cope with SIP reliability and security. It benefits from the advantages of both server-based and P2P-based SIP networking. We successfully validated the functionality of CoSIP on local testbeds as well as on PlanetLab. We also integrated our CoSIP implementation with our experiments on virtualization. Finally, our experiments validate the $D^2R^2 + DR$ strategy, notably the inner control loop.

## 5 Building Resilient Services using Virtualization

### 5.1 Abstract

OS virtualization techniques such Xen and Kernel-based Virtual Machine (KVM) can be used to provide an abstraction from the underlying hardware resource and react efficiently to challenges, such as overload or hardware failure. In this section, we summarize our experiments and results in the area of virtualization with respect to service resilience.

### 5.2 Methodology

#### 5.2.1 Virtualization as a Resilience Mechanism

This experimentation use-case investigates the benefits of virtualisation for resilient services. In the attached paper [FFCdM11], we provide an extensive analysis of how to use virtualization in the context of the $D^2R^2 + DR$ strategy. For example, we identified the distribution of VM snapshots in different locations in the network as a crucial Defence mechanism. As a remediation mechanism, these snapshots can be started and made available within a short timeframe, e.g., a couple of seconds, when they are needed. This can be the case, e.g., upon an increasing service load, or to replace a failing service.

We identified VM replication and VM migration as frequently useful remediation mecha-

nisms. In the ideal case, VM replication or migration should occur with respect to a previous snapshot as well. This allows for transferring only a delta of the VM state over the network, in order to reduce the required bandwidth consumption, and the time required to achieve the remediation. For example, the *libguestfs*[6] provides a set of tools which can be executed at a physical host to view and edit files within a guest VM.

### 5.2.2 (Wide-Area) VM Live Migration

**VM Live Migration**

VM *live* migration allows for migrating the hot state of the VM, i.e., main memory, cache and CPU registers from a source physical host to a target physical host. A VM is typically migrated within the same layer-2 broadcast domain. Thus, the IP address of the VM remains the same (See [FFCdM11] for more details how the VM can keep its IP address). TCP connections do not break. This makes the migration transparent to a client connected to the VM, and also to the user who is, e.g., in the middle of an Online Shopping session.

Technically speaking, a VM live migration is realized as follows (see also Figure 13):

1. Resources are allocated for the VM at the target physical host.
2. The hot state of the VM is copied iteratively from the source physical host to the target physical host. At each iteration, only the memory pages which have changed since the last iteration (also called "dirty pages") are copied.
3. The last step is performed iteratively until the number of dirty pages is sufficiently small.
4. Subsequently, the VM is stopped, and the rest of the hot state is copied.
5. Finally, the VM is resumed at the target host. Connectivity is re-established by announcing the IP address of the VM in the new layer-2 broadcast domain.

Since the VM can continue running during the iterative copy procedure, i.e., until step 3. above, the live migration results into a small downtime (the time between step 4. and step 5.), e.g., less than 100 ms [CFH+05].

**Wide-Area VM Live Migration**

In [FFCdM11], we identified cases, where a VM needs to be migrated to a different data centre in a different geographic location. For example, a natural disaster, e.g., tsunami or hurricane, is expected to hit a data centre within the next hours. Particularly in these cases, services like news, video streaming and Voice over IP (VoIP) are critical to humans in order to get the latest information, be better prepared for the natural disaster and contact their families and friends.

In case a VM is migrated to a different geographic location including its hot state, one talks about *wide-area VM migration*. Looking at the technical details of a wide-area migration, and comparing to the *local-area* live migration described above, we identify a *mobility* problem: the VM is resumed within a different network. The network configuration of the VM when the migration is achieved, i.e., IP address, route and eventually ARP cache and other configuration parameters, are not compatible with the new network. The solution can be dissected into the following steps:

[6] `http://libguestfs.org/`

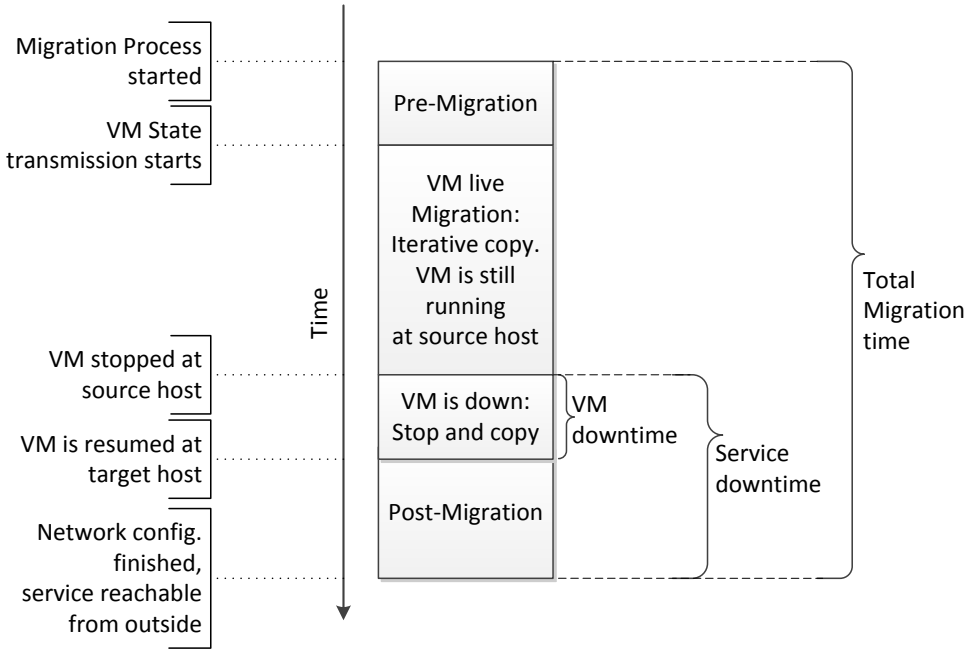



Figure 13: Wide-Area VM Migration Phases.

1. **Network configuration:** The VM needs to acquire its new network configuration, notably new IP address and routing information. This step allows for basic IP connectivity to the VM which can be tested, e.g., via `ping`

2. **Service recovery:** A service running within the VM needs to become available to its client again. This can not happen before the "Network configuration" step.

In fact, we notice here that the actual goal is to keep the service downtime as low as possible and that

$$serviceDowntime \geq VMDowntime \quad (1)$$

The $2^{nd}$ step itself, "Service recovery", can be dissected into the following steps.

**2.1 Service announcement:** The new location of services running within the VM needs to be propagated, e.g., via DNS updates, in order to be reachable for new incoming connections.

**2.2 Service reconnectivity:** The connectivity between a service running with the VM and its current clients (i.e., which are already connected) needs to be recovered.

Figure 13 provides a high level overview of the VM live migration phases. The difference between a wide-area live migration and local live migration is essentially in the last phase, i.e., "post migration".

Now, we describe different options for implementing the steps, 1, 2.1, and 2.2 above.

**Network Configuration**

In the case of a local migration, the hypervisor at the target host sends an ARP update message in the layer-2 broadcast domain with the IP address of the VM. Once the ARP

message is propagated, all layer-2 traffic with the VM IP address as destination is forwarded to the target physical host instead of the source physical host. The target physical host forwards the traffic then to the VM (see also [FFCdM11])[7]. Given that the VM keeps its IP address, the migration is transparent to the client.

In the case of a wide-area migration, the VM is resumed in the target network including its hot state, i.e., RAM, cache and CPU registers. However, the network configuration of the VM is not compatible with the new network. This is similar to the case where a physical machine with an Ethernet interface, e.g., `eth0` is:

- Connected to a network $A$ via `eth0`,
- Physically disconnected (example by unplugging the Ethernet cable),
- And then reconnected to a physical network $B$.

Modern operating systems today allow for detecting the unplugging and plugging event. Typically, the network interface (NIC) driver provides this information to the operating system. Thus, when the VM is disconnected from $A$, the NIC loses its IP address. When the NIC is reconnected, DHCP is triggered (assuming that the NIC is configured with DHCP) and the NIC acquires a new IP address, and possibly new route configuration.

Thus, two questions arise here:

- Does the VM notice that it has been disconnected and reconnected to another network as it is the case for a physical machine and physical NIC described above?
- Can this network configuration change, including DHCP request and response, be performed within the anticipated downtime of, e.g., $<$ 100ms?

For the first question, given the lack of a physical event, there is a need for a notification to the VM that the network has changed. As for the second question, acquiring IP and network configuration via DHCP can take easily several seconds.

During our experimentation, we have been investigating different options to notify the VM about the network change.

- The first (somehow simple but naive) option would be to configure the VM to restart DHCP regularly, e.g., every 10s. However, this solution may incur a downtime of more than 10s (or whatever time interval is configured. An interval of one second would be still too high since we anticipate a downtime in the order of 100ms).
- We experimented also with IPv6 where the physical host sends route advertisement regularly. This option, however, has the same problem as DHCP above.

After these experiments, our goal was to find an asynchronous solution where the network configuration is triggered as soon as the migration is successfully achieved, instead of a solution where the network configuration procedure is triggered at regular intervals.

[7] Traffic between the VM and the physical host is forwarded via a virtual bridge running inside the physical host and connecting the IP stack of the physical host with the IP stack of the VM. Virtualization solutions usually offer also the possibility to run the VM behind a virtual NAT as well. However, this is not typical for a VM which hosting services that should be reachable from outside, given the additional complexity that would be required for NAT traversal.

- In the simplest case, the virtualization API would allow for opening a shell in the VM and sending a command through it. Thus, the notification about the network change, together with the new network configuration parameters, can be provided through this shell to the VM after migration. Manipulation of the VM from outside (i.e., from the hypervisor or the physical host) is currently continuously under further development (see, e.g., the `libguestfs` mentioned above). Thus, we expect this feature to appear in the future.
- Using events, e.g., ACPI, with a dedicated code, the VM may trigger the network reconfiguration upon receipt of this event.
- A Configuration Server running within the VM expects configuration messages from the physical host. Communication with the configuration server takes place using TCP/IP.

The last option might sound odd at a first glance. Particularly, the question remains how to reach the VM via TCP/IP before the VM acquires its new IP configuration. Nevertheless, we ended up by implementing this solution. Nevertheless, any other option to communicate between the physical host and the guest VM which allows for the notification about the changes in the network configuration and does not incur a significant additional delay can be used. In any case, we provide some details about our solution with the Configuration Server for the sake of completeness here:

**Configuration Server**

We configured a 2nd network interface for the VM during the migration phase, e.g., `eth1` with the new IP address of the VM. Upon this configuration, an entry is automatically added to the VM routing table for the layer-2 broadcast domain for `eth1`. When the migration is finished, the VM is resumed at the target host. Since the target physical host is in the same broadcast domain (via the virtual bridge), the host and the VM can already communicate without any problem. The only configuration step which is still required is to update the default gateway for the VM in order to be able to communicate outside its layer-2 broadcast domain and be reachable for clients in the Internet. Thus, the target physical host sends an XML-RPC `setup` message to a Configuration Server (which we implemented) running within the VM with the new network configuration. The Configuration Server can apply the new network configuration, notably its default gateway, and becomes reachable via the new IP address also from outside. Figure 14 provides an overview of this procedure.

### 2.1 Service announcement

A service is typically announced using DNS. When a VM is migrated and acquires a new IP address, an update of the DNS records is required. Unfortunately, DNS records with a time-to-live (TTL) of up to 24 hours are not rare. One defence mechanism is to announce DNS records with a short TTL, e.g., 60 seconds. Unfortunately, not all DNS clients honor the DNS TTL. Particularly, many ISPs increase the DNS TTL artificially in order to decrease the traffic on their sides. This is one of the reasons why one should not rely fully on DNS to update the service location and deploy further update mechanisms as explained below.

### 2.2 Service Reconnectivity

After the VM has acquired its IP configuration, it needs to recover its connectivity to the clients. This is represented by the `update` message in Figure 14. However, this update can

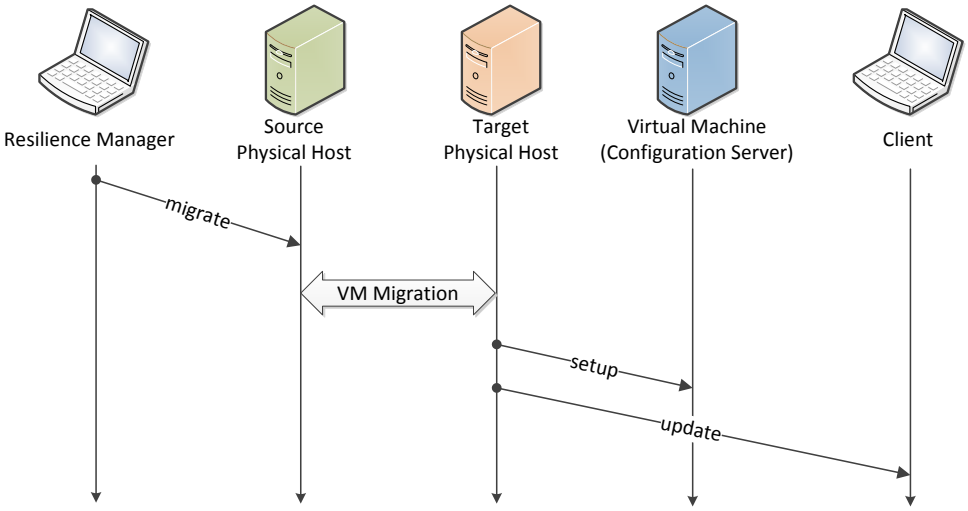



Figure 14: Wide-Area VM Migration: High Level Message Flow

Table 1: Service Re-Connectivity Solutions

| Layer | Through indirection | Through End-to-End |
|---|---|---|
| Application | Proxy | SIP, XMPP, web sockets |
| Network | MIP with Home Agent, NAT, Layer 3 gateway | MIP Route Optimization |
| Data link | Layer 2 gateway | N2N |

be performed at different layers, and either end-to-end or via an indirection point. Table 1 provides an overview of different solutions which come into question in order to address this problem. In [FFCdM11], we discuss these options more in details. In Section 5.3 below, we describe our experiments with location update mechanisms at different layers and compare the solutions.

### 5.2.3 Relationshp to the $D^2R^2 + DR$ strategy

In this section, we describe how the $D^2R^2 + DR$ strategy can be validated using our virtualization concepts and experiments.

**Defence** Summarizing the defence mechanisms described above, we can say that

- VM snapshots should be distributed beforehand in different locations in the network.
- Using DNS with short TTLs should be deployed beforehand in order to reduce downtime when a service is migrated.

**Challenge Emulation** We emulate a challenge where a live migration remains possible. As explained in [FFCdM11], this depends on whether a failure is predictable. If a failure is not predictable, the hot state of the VM, i.e., main memory, cache and CPU registers will not be available during the remediation. Thus, live migration as a remediation mechanism is not feasible. However, using virtualisation, a previous snapshot of the VM image can be started and made available.

**Detection** No detection has been deployed in this experimentation use-case. We assume that there are appropriate mechanisms put in place that detect the challenge (for example, the Chronicle Recognition System (CRS) as mentioned below in this deliverable) and identify VM live migration as an appropriate remediation mechanism as well as which target physical host the VM should be migrated to.

**Remediation** The remediation mechanism consists of the VM life migration and the re-connectivity to the clients.

**Recovery** The situation after the VM migration might be sub-optimal due to different reasons. For example,

- Economic reasons: after the remediation, the VM might be running on the infrastructure of a third party provider, which may lead to high costs.
- Technical reasons: the service provided by the VM after successful migration might still be sub-optimal. For example, the new location of the VM might lead to a high latency. Or the new allocation of the VMs to the physical resources leads to sub-optimal performance.

The recovery mechanism can be, e.g., the migration of the VM back when the challenge ceases, or at least the migration to yet another location where *i)* better performance can be provided, *ii)* the VM deployment does not incur high costs, *iii)* or any other reason.

## 5.3 Experimentation and Results

After explaining how virtualization can be used as a resilience mechanism at the different phases of the the $D^2R^2 + DR$ strategy (Section 5.2.3), and after explaining the difference between live migration and wide-area live migration, and the technical difficulties related to wide-area live migration (Section 5.2.2), we now describe our experiments which address wide-area live migration given that this can be considered as a challenging scenario. We describe two different experiments. The first (Section 5.3.1) uses end-to-end mobility at the application layer for notifying clients about the new service location. The second (Section 5.3.2) uses a layer-2 tunnel which is re-directed to the new VM location upon migration.

### 5.3.1 Wide-Area VM Migration with Application-Layer E2E Notification

Looking at Table 1, while end-to-end solutions have the clear advantage that they do not require additional infrastructure, they are typically intrusive solutions and require major modifications to the client. For example, Mobile IP requires support by the client. N2N [DA08] needs the

client to join a layer-2 P2P VPN and be in the same VPN as the VM in order to receive VM's location updates. At the application layer, using for example, SIP or XMPP requires a modification of the application to support mobility. However, in case of web-based applications, the new web sockets [8] standard can be very helpful as we will see below using web-based video streaming as an example application.

**Application** We chose to implement wide-area VM migration with an example application: web-based video streaming. The reasons are as follows:

- We argued that natural disasters are one use-case where such a wide-area migration can be very helpful. In this case, video streaming is a critical application to provide updated news.
- Video streaming is an application where long downtimes will be visible to the user. Thus, our goal to achieve a low service downtime, e.g., $<$ 100ms fit well with the goal to make the migration as transparent as possible to the user.
- We chose a web-based implementation, given the increasing popularity of web-based services (See, e.g., Chrome OS or Web OS).
- Using a web-based application, we can benefit from the newest development in this area, e.g., the web sockets standard as explained below.

**Using Web Sockets for Asynchronous Notifications** Since HTTP is a request/response protocol which does not allow asynchronous notification from the server to the client by default, we use the new *web sockets* standard. Web sockets are an emulation of TCP sockets at the application layer. Thus, they provide reliable transport and guarantee the right order of message delivery. In contrast to TCP, communication is frame-based where each message is marked at the end with a delimiter. Communication in a web socket connection is bidirectional. Thus, either client, e.g., Internet browser, or server can send a message anytime.

Web sockets can be used within Javascript code running in the Internet browser. Since the Javascript code is downloaded from the web server at the beginning of a video session, there is no need to modify the client. This is a major advantage compared to, e.g., Mobile IP. All the instructions required for the client on how to recover the connectivity to the VM upon wide-area migration are provided in the Javascript code.

**Experiments** We implemented wide-area VM live migration as described above. We used web sockets and Javascript for notifying the clients about the new service IP address and port number after migration. Figure 15 provides an overview of our testbed. A system administrator can verify the availability of the service at different locations of the network via a web interface. Furthermore, (s)he can trigger a wide-area migration of a VM[9]. On the other hand, a user is watching a video delivered by the service.

We implemented

- The Migration Servers running at each physical host

[8] http://websocket.org/

[9] Certainly in the absence of the system administrator, the service monitoring, challenges detection and the migration triggering might be automated

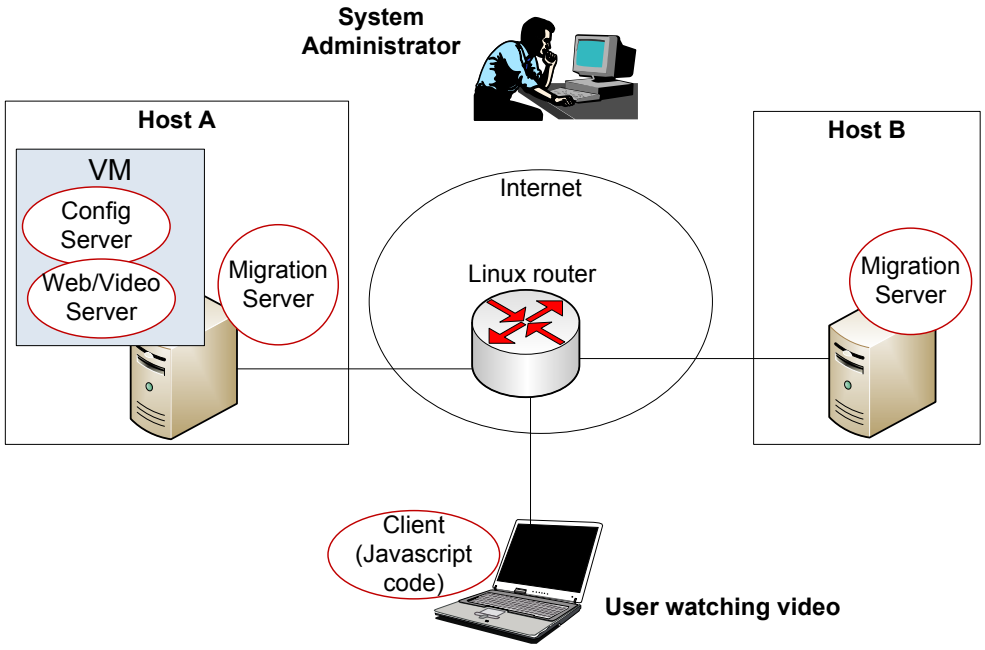



Figure 15: Testbed for Wide-Area VM Migration with Application Layer Notification

- The Configuration Server running within the VM which receives notification from any of the migration servers about upcoming or completed migration and if necessary applies modification to the network configuration of the VM
- The Web Server running within the VM which is informed by the Configuration Server about upcoming or completed migration. The Web Server also provides the client with the necessary information to remain connected even after migration. Furthermore, the Web Server provides the actual application, which is a list of videos that the user can choose from, and the video streaming once the user has made her choice.
- The Javascript code running at the browser. This code is downloaded from the web server at the initial connection. This code enables the reconnectivity to the VM upon migration.

**Results** In our experiments, we carried out several instances of our wide-area VM migration scenario and achieved a service downtime in the order of only 400ms. At early experiments, we experienced a service downtime of 30s or more. Thus, we realized that several optimizations were needed in order to reduce the downtime as much as possible. For example, the VM can be additionally configured with its new IP address during the migration phase already while the previous IP address is still active. Moreover, some parameters have to be adjusted, e.g., timeouts for ARP cache as well as *forward delay* for the bridges. When the standard Linux bridging is used on the VM hosts, it is important to ensure that the forward delay for the bridges is set to 0. This parameter is set to 15 seconds by default, causing a new port to go into learning state for 15 seconds when a new port is attached to it before going into forwarding state. Note however, that we stumbled across these problems with ARP and forward delay when doing a local-area VM migration as well, i.e., with an Ethernet switch instead of the router in the center of the network.

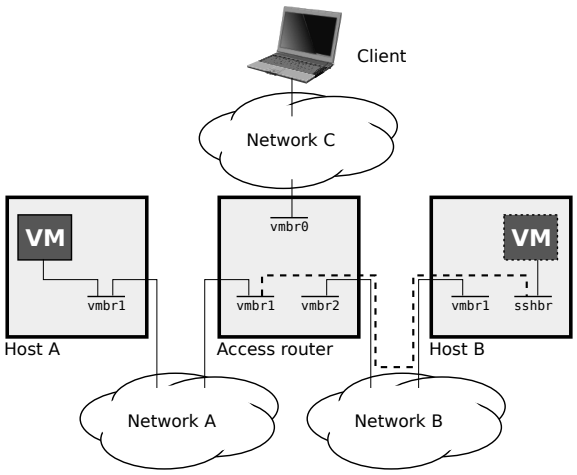



Figure 16: Migration setup for layer-2 tunneling

### 5.3.2 Wide-Area VM Migration with Layer-2 Tunneling

When performing network redirection at the Application Layer, as shown in the previous section, one loses transparency of the approach. In particular, the application has to be aware of the migration itself and support it to a certain extent (e.g., by using a protocol that supports service mobility). On the other hand, service providers are likely to have legacy services in operation that can not be changed easily. To those services, migration has to be fully transparent. Thus, we also investigate VM migration with layer-2 tunneling.

**Application** In this experiments, we were mainly interested in the downtime a certain service will experience during migration. This downtime should be minimized in order to avoid service interruption for customers. The migration itself will be affected primarily by the amount of state change the service causes. The more state changes a service causes during migration, the more often state has to be updated at the target host. Thus, we investigated two service variants:

- A service with low activity (effectively a VM with only background processes running)
- A diabolical service with a high amount of state changes (implemented by requesting high amounts of RAM within the VM and repeatedly filling it with random numbers)

**Interconnecting the uplink router with the target host** In order to properly redirect traffic after the migration, the uplink router of the initial host the VM resides on takes the responsibility of redirecting network traffic to the target host which will run the VM after the migration. This is depicted in Figure 16.

In this scenario, we assume that the VM is initially residing on Host A. Host A is endangered by some threat or challenge, causing a need for the VM to be migrated to a different host (Host B). Host A is connected to the internet via an access router. It is assumed that this access router can be used as an indirection point to redirect traffic after migration. Host A and Host B are located in different networks, causing a wide-area migration. A client is also set up to simulate live access during the migration.

The migration itself is performed by creating a virtual link between the access router and Host B. This ensures that – logically – the VM stays in the same network all the time. In particular, a tunnel is created between the virtual bridges *vmbr1* on the access router and *sshbr* on Host B. This is depicted by the dashed line in the picture. Once the tunnel has been set up, both, cold and hot state are transferred to Host B. They are then synchronized continuously with the original VM at Host A until the difference is negligible. At this point, the original VM is terminated and the copy on Host B starts to take over operation. The access router will then forward all future traffic of the clients to the VM on Host B.

**Experiments** We performed experiments with the setup described above. In particular, we were interested in the downtime of a service during live-migration of a VM. A user accesses the service and is transparently redirected once the VM changes location. During our experiments, it became clear that measuring the downtime from a user perspective (i.e., from the Client in Figure 16) was too coarse grained for our purposes. This is due to random delays introduced by the networks between the Client and the VM. Instead we monitored network connectivity of the VM directly on Host A.

**Results** The total downtime of the live migration amounted to about 600 milliseconds. This is significantly better than any cold migration approach and about the same order of magnitude as a live migration within a single subnet. We were able to show that live migration across wide-area networks is feasible and can be used to counter certain challenges. By using layer-2 tunneling techniques, the migration process itself is fully transparent to the service within the VM. This allows for legacy services to make use of VM mobility without having to change the service itself.

## 5.4 Conclusions and Lessons Learned

In our experiments, we came to the conclusion that wide-area VM migration can be performed with relatively low downtimes. Network redirection can be performed with a variety of means, without significant impact on the downtime of the virtual service. In particular, we were able to see that network redirection on the application layer and network redirection on the link layer achieved results of the same order of magnitude with regard to downtime. When applying wide-area migration as resilience mechanism, one can therefore choose the approach which fits the application best.

We noted two more interesting aspects during our experiments. First, wide-area migration can be performed even if the cold state has to be migrated as well. While this will prolong the total migration process, the actual downtime of the virtual service can be kept independent of the amount of cold state to transfer, provided the hypervisor offers means to synchronize cold state (as was the case with KVM). If the hypervisor doesn't support explicit synchronization of cold state (as is the case with XEN), then the downtime of the VM will increase, as the cold state has to be synchronized manually (a process during which the VM has to be paused).

Second, very frequent changes to the hot state (i.e., provoking high volatility of the hot state via a high number of forced state changes) prolongs the migration but has no significant effect on the downtime of a service. In our case, we forced a high number of updates of the RAM of the VM, by repeatedly filling memory with random numbers. This caused the total migration time to increase, but the actual downtime of the VM was similar to our previous experiments.

The results gained here are highly relevant for the upcoming cloud computing paradigm. Cloud computing uses resource virtualization as a basic building block. We showed that those virtualized resources can be migrated freely, even beyond network borders, in order to react to challenges. This can be used as a basic resilience mechanism in clouds.

# 6 Publish-Subscribe Platform

## 6.1 Abstract

We have proposed in [FFH+11] a service resilience framework based on risk analysis, threat detection, and policy-based reaction. The work realized in this part of WP4 consists of applying the threat detection phase of this framework in the context of a publish-subscribe (PubSub) platform.

This section will first focus on the context of the experimentation. A short description of the threat detection part of our service resilience framework is given, followed by a presentation of the PubSub paradigm [Bir93]. The basic tools needed for our experimentation are then presented: (i) Nagios, an open source monitoring software; (ii) SolAdvisor, a simulation programme from Solent, the platform equipments' manufacturer; (iii) Chronicle Recognition System, an event correlation tool developed in France Telecom and used to produce the alerts starting from basic alarms transmitted by Nagios.

The second part of the section describes the PubSub platform, consisting principally of two XML routers, on which this work is based. Different security-related resilience issues have been identified for this platform: sniffing/scanning attacks, session hijacking, DoS. This deliverable will focus on flooding attacks. The experimentation's implementation, accompanied with some preliminary results, is then sketched. We will finally draw some conclusions and lessons learned of this work at the end of the section.

## 6.2 Methodology

### 6.2.1 Service resilience framework

The use of telecommunication services has become an essential part of our daily life (voice communications, videoconferencing, e-mail, banking operations, e-commerce, ) and a requirement for the majority of administrative operations. This dependency is disturbing because the amount of threats surrounding these services is ever increasing (DoS, viruses, eavesdropping, accidental failures, operational faults, etc.).

To render these services resilient in operation, the service resilience framework we have proposed is based on 3 principal steps [FFH+11]:

- the starting point is a risk analysis allowing identification of threats that might perturb a service - this operation results in a prioritized list of risks and the appropriate actions to be undertaken when these appear;
- detection techniques with the suitable parameters, found thanks to the previous phase, are then deployed to identify the threats in real-time - detection is performed through

two phases: (i) monitoring for collecting data, detecting anomalies when these happen and generating alarms; (ii) alarms correlation to retrieve relevant data and detect the presence of serious threats;

- when such a threat is reported, an alert is sent to a policy-based reaction engine - we have chosen a context-aware reaction engine allowing the change of configuration when a threat context appears without interrupting the service.

The detection is a twofold task. Services first need to be monitored continuously in order to identify any abnormal situation when it appears. Thus, our work towards threat detection begins with the service monitoring aspect: the goal is to trigger alarms when anomalies happen. These alarms then need to be analyzed and correlated to identify the presence of a real challenge, or not. The idea is to condense alarms to retrieve just relevant data.

Numerous paradigms have been proposed upon which event correlation techniques are based. These approaches derive from different areas of computer science: artificial intelligence (rule-based systems, neural networks, model-based systems, decision trees, case-based systems), fault propagation models (code-based techniques, dependency graphs, Bayesian networks, causality graphs, phase structured grammars), and model traversing techniques using object-oriented representation of the system.

Among these various techniques based on different aspects of correlation such as topological, functional and time aspects, we focus here on the temporal scenario recognition, through the use of the Chronicle Recognition System (CRS), developed in France Telecom.

### 6.2.2 The PubSub paradigm

A pioneering work on the subject of publishing/subscribing to messages is the research on ISIS, a system providing tools to support the construction of reliable distributed software [Bir93]. Brokerage and trading systems were the use case illustrations for this type of paradigm, i.e., integrating large numbers of demanding applications and requiring timely reaction to high volumes of pricing and trading information.

Applied for application integration in diverse domains such as financial, process automation, transportation, etc., this paradigm is efficient to implement for interconnecting applications in a distributed environment. PubSub systems contain information providers, who publish events to the system, and information consumers, who subscribe to particular categories of events within the system, which ensures the timely delivery of published events to all interested subscribers. A PubSub system also contains message brokers responsible for routing messages between publishers and subscribers [BCM+99].

The first of these systems were *subject-based*, i.e., each unit of information (event) is classified as belonging to one of a fixed set of subjects (a.k.a. groups, channels, or topics). Publishers are required to label each event with a subject; consumers subscribe to all the events within a particular subject. In the example of stock trading, one may define a group for each stock issue; publishers may post information to the appropriate group, and subscribers to this group will receive information regarding the referenced stock.

An alternative to subject-based systems are *content-based* subscription systems which support a number of information spaces, each associated with an event schema (see below) defining the type of information contained in each event. The stock trade example may be characterized as a single information space with an event schema defined as the tuple [issue:

string, price: euro, volume: integer]. A content-based subscription is a predicate against the event schema of an information space, such as (issue="Philips" & price $<$ 12 & volume $>$ 1000).

In this context of message-oriented routing, PubSub systems have their sources (senders) labeling each message with the name of a topic, rather than addressing it to specific recipients. The message is then sent to all eligible systems having asked to receive messages on that topic. This form of asynchronous architecture is more scalable than point-to-point alternatives such as message queuing, since message senders need only concern themselves with creating the original message, and leave the task of servicing recipients to the messaging infrastructure. It is a very loosely coupled architecture, in which senders often do not even know who their subscribers are.

Some applications of this paradigm, e.g., stock exchange ones, are very demanding in terms of resilience. Besides traditional dependability concerns such as high availability (service continuity has to be guaranteed against hardware and software failures, or DDoS attacks), the data reliability, i.e., its integrity, need to be insured through the assessment of the publishers identity. The messages' confidentiality is also part of the security requirements, i.e., data interception, or usurpation of legal subscribers identity, are part of high priority challenges.

### 6.2.3 Nagios

Nagios (Nagios Ain't Gonna Insist On Sainthood) is a popular open source network monitoring software application running on Linux [NAGa]. It watches hosts and services, alerting users when things go wrong and again when they get better. Nagios' monitoring function applies to a wide spectrum of items [NAGb]:

- network services (SMTP, POP3, SNMP, FTP);
- host resources (processor load, disk usage, system logs);
- anything else like probes (temperature, alarms, ...) which have the ability to send collected data via a network to specifically written plugins.

The software can provide monitoring through remotely-run scripts via Nagios Remote Plugin Executor , and support SSH or SSL encrypted tunnels. These custom scripts allow monitoring of in-house applications, services, and systems. A variety of plugins are available in this environment, e.g., plugins for graphing of data. Designing plugins is facilitated, allowing users to easily develop their own service checks depending on needs, by using the tools of choice (C++, Perl, Python, PHP, etc.), thus permitting an extendable architecture for integration with in-house and third-party applications.

The users are able to define a network host hierarchy using "parent" hosts, allowing detection of, and distinction between, hosts that are down and those that are unreachable. Concerning alarm notifications, there is a possibility for contact when service or host problems occur and get resolved (via e-mail, SMS, or any user-defined method through plugin system). The reporting ensures SLAs are being met, and provides historical records of outages, notifications, and alert response for later analysis.

Nagios provides its users the possibility to define event handlers to be run during service or host events for proactive problem resolution, and the support for implementing redundant monitoring hosts. It is noteworthy that data storage is done in text files rather than database,

and log file rotation is performed automatically. Finally, a Web interface can be supplied for viewing current network status, notifications, problem history, log files, etc. This is a multi-user access which allows stakeholders to view, e.g., infrastructure status. In addition, user-specific views ensure clients see only the infrastructure components they are authorized for. In this experimentation, we use Nagios to collect alarms sent by SNMP when predefined thresholds are exceeded: the monitored parameters are bandwidth and disk utilization.

### 6.2.4 SolAdvisor

SolAdvisor is a reference publisher/subscriber application for the Solace Messaging Platform [Sys10]. It is a Java-based desktop application, supported on computers running Windows XP. SolAdvisor allows the user to test the Solace router network by doing the following:

- managing publishers and subscribers;
- performing Publish and Subscribe tasks;
- monitoring router activity.

To deliver the first functionality, SolAdvisor provides an interactive, responsive, and user-friendly graphical user interface for publishing and subscribing using existing publishers and subscribers in a Solace Messaging Platform. In addition, a 'Configuration' view can be used to create time-saving shortcuts for different network objects, easing the management of various Publish/Subscribe tasks.

The application also allows to test how Solace routers handle various Publish/Subscribe tasks. In SolAdvisor, publishers can be configured to publish messages, with or without meta-data or binary attachments, over TCP connections. Subscribers can manage their subscriptions to receive these messages.

Finally, for the monitoring router activity, once the user has started publishing and subscribing to XML messages, SolAdvisor immediately begins collecting and displaying statistics. The statistics collected appear in a tab in the 'Individual Publisher and Subscriber Details' areas. In addition, it is possible to enable logging to view information on all Publish/Subscribe events. SolAdvisor is exploited in this work to manage publishers and subscribers by creating their accounts, and, for each publisher, by initiating a set of messages to send to the platform in order to simulate a flooding attack.

### 6.2.5 Chronicle Recognition System

CRS is an event correlation tool based on symptom-to-fault knowledge represented in form of chronicles which are a specific kind of rules composed of temporally constrained events [CD00]. In other terms, a chronicle is a set of events, happening at different time points interlinked by time constraints, and whose occurrence may depend on the context. Each event may be in different states. An example of chronicles is represented in Fig. 17 where time points t1, t2, t3, t4 are partially sequenced and the time interval between t1 and t2 needs to be comprised between 2 and 5 minutes. On the other hand, there is no constraint regarding transitions such as t2 to t4.

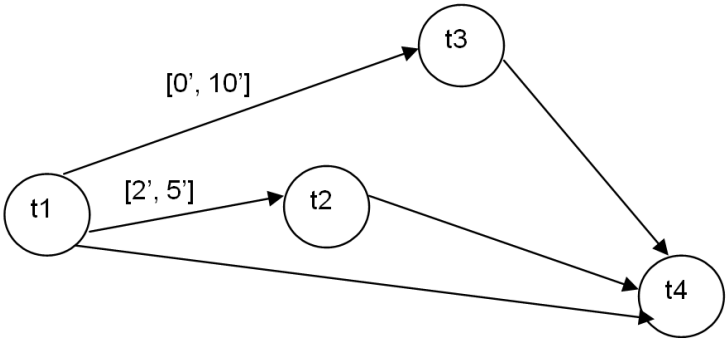



Figure 17: A chronicle sample

As sketched earlier, events are represented by their name, state/transition (from one state to another) and time/interval of occurrence. Time information enables to sequence events, and even to specify time spans, between their occurrences. The example mentioned hereafter provides an insight about the language used to write chronicles. This chronicle is composed of four events named B, L, IPloss and link. Each one of these events is in a certain state, or is transiting from one state to another:

- the event link is in the WiFi state between the two time points t1 and t4;
- the event L is transiting from the state 'good' to the state 'bad';
- as '?' means any state, the event B, resp. IPloss, needs to be in transit from any state to the state 'critical', resp. 'high'.

*Chronicle myChronicle*{
*event (B:(?, critical), t1)*
*event (L:(good, bad), t2)*
*event (IPloss:(?, high), t3)*
*hold (link:WiFi, (t1, t4))*
*t2 - t1 in [2',5']*
$t2 <$*t4*
*t3 - t1 in [0',10']*
$t3 <$*t4* }

Defining chronicles and updating the chronicle database represent a big challenge with CRS. The definition of chronicles is a task to be done either by an expert, or using an automatic learning engine. One of the most important advantages with CRS is that it combines an on-line symptom-based correlation approach, and an off-line model-based chronicle learning

approach. The on-line approach allows detection of challenges in a timely frame, while the off-line method, based on a model to detect any abnormal behavior in the system, generates new chronicles to detect the next appearance of this misbehavior. Thus, the off-line approach allows for automatic acquisition of the chronicle base from the model if this latter is a progressive and powerful model. In our experimentation, the on-line approach is used for detecting flooding attacks, i.e., chronicles are defined on an expertise basis.

## 6.3 Experimentation

### 6.3.1 FT's PubSub platform

This PubSub platform, hosted in Orange Labs (Sophia-Antipolis, France) is composed of two Solace Systems XML routers (Xsol3, Xsol4) accessible through proprietary interfaces via SSH, SFTP, or SNMP (Fig. 18). Two APIs (Java and JMS) are used for message publishing and reception. A Web server, running on a Linux PC with the use of open source software (Tomcat, MySQL, Java), is exploited for managing a database describing the network configuration and its users. A user portal offering HMI for a set of services is also provided by this server.

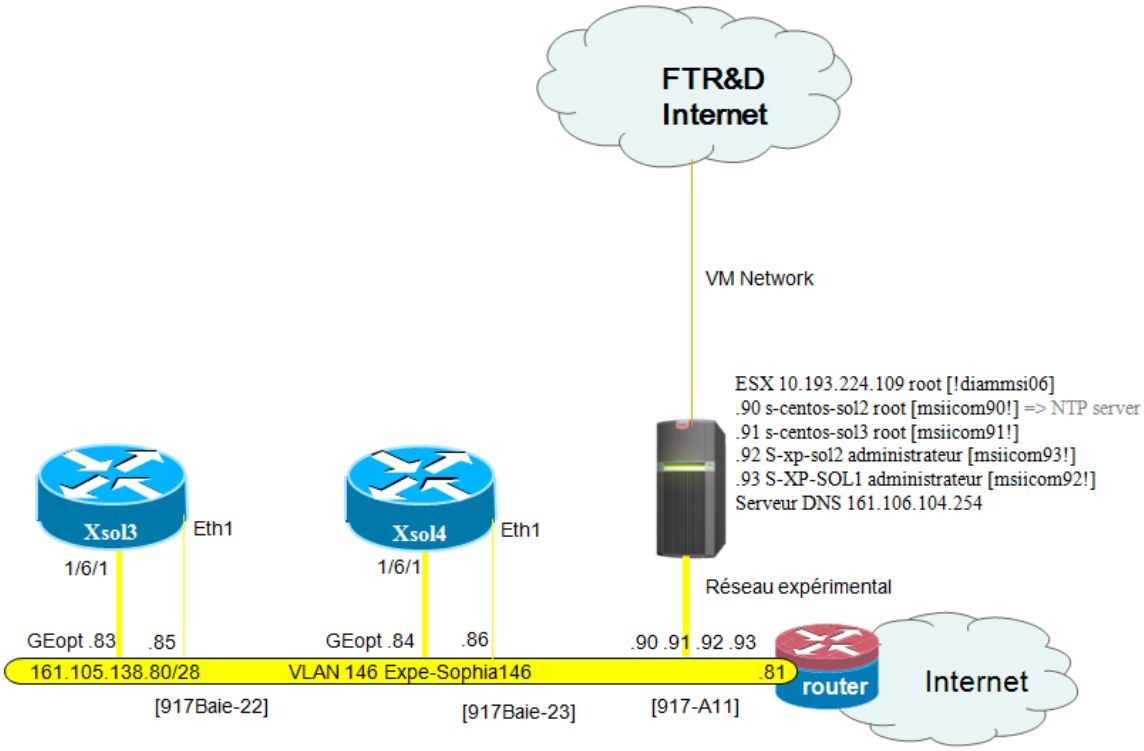



Figure 18: Global architecture of the PubSub platform

The filtering functions and routing information allow to decouple message senders and

recipients. This platform, based on XML routers using hardware to process messages, allows very high performance; the network covers itself a network of (traditional) IP routers. Through this platform:

- content-based routing is done from selection filters (subscriptions) of XML documents;
- two types of users (data publisher/sender, data subscriber/client) can be managed;
- publishers broadcast their messages to authorized recipients;
- subscribers only receive relevant data, thanks to contents filtering (subject, ...), through chosen communication channels (RSS, SMTP, SMS, ...);
- messages are stored until they have been issued;
- in addition to access control for both publishers and subscribers, a Web portal manages the administrative tasks (client configuration management, accounts provisioning, ).

The PubSub platform includes security-related resilience issues such as:

- spying-like attacks aiming to break confidentiality, e.g., sniffing and scanning attacks;
- session hijacking intending to tamper the messages integrity, exploiting authentication weaknesses;
- denial of service which are numerous and based on several techniques, e.g., flooding attacks which increase networks traffic and load, resulting in service performance deterioration, reaching then the service denial, i.e., its reliability and availability.

Confidentiality, integrity, authentication, availability and accountability must be ensured to protect data against eavesdropping, modification and damage. To render the PubSub platform resilient, the first step is to name relevant challenges; experts are then required to identify what to use for monitoring, detection and remediation. For the purpose of this deliverable, we will focus on flooding attacks. The goal of the experimentation is to detect, among others, flooding attacks using the described detection framework described in [FFH$^+$11] .

For surveillance purposes, we are exploiting the SNMP protocol to invoke the PubSub platform and to get available data, e.g., number of connections, messages publication/delivery rate, disk utilization, and other parameters.

Our purpose is to calculate the PubSub service's occupation rate. Actually, if the publication process increases while the messages delivery is halted or reduced, e.g., due to a few number of subscribers on-line, the disk - used to store messages waiting for a delivery - utilization necessary increases because the platform contract guarantees that not a single message is lost.

In a regular functioning scheme, it does not constitute a problem, because resources dimensioning is realized in such a way that there is enough space to host all messages for which delivery is pending. This dimensioning process has its limits, i.e., the disk hosting messages is of finite size: if the number of published messages is close to the theoretic maximum allowed by this disk's size, under certain conditions, saturation can be reached.

Moreover, in the case of a flooding attack, all dimensioning efforts are ruled out because no one can predict the size of false data published unwittingly by registered publishers who

are controlled by attackers, or by unregistered publishers (attackers themselves) who gained access to the platform. In these cases, if the phenomenon reaches a certain threshold, storage resources will be saturated leading to authentic messages lost. The intention is thus to detect these situations.

To do so, we define two thresholds TI and TE for the publication rate (ingress message rate, IMR), and the delivery rate (egress message rate, EMR) such that:

- if $IMR <$TI, resources dimensioning can always deal with the message storage aspect whatever the delivery rate;
- if $IMR >$TI, we may be in a flooding attack situation - this could also correspond to a situation with abnormal peaks of legitimate traffic, e.g., software failure leading to an avalanche of messages
  - if $EMR <$TE, a high ingress message rate is coupled with a small delivery rate (e.g., nearly no subscriber logged in for some exceptional reasons like at night), the saturation can be reached, i.e., the platform will soon become unavailable. It is thus the challenge scenario that we need to detect.
  - if $EMR >$TE, the delivery rate is comfortable enough, allowing the platform to face high ingress message rate

The correlation task targets the detection of such challenge scenario performed by a set of publishers trying to overwhelm, intentionally or not, the PubSub platform, denying thus the service for the legitimate users. It is based on the chronicle in (Fig. 19).

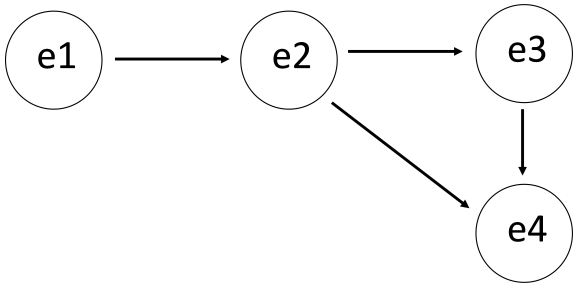



Figure 19: Chronicle for detecting a flooding attack

The symbols e1, e2, e3, e4 represent the following events:

- e1: high number of connections;
- e2 : ingress message rate higher than the threshold;
- e3 : egress message rate lower than the threshold;
- e4: high disk utilization.

This chronicle shows that bandwidth and disk used by the publishers are very high and exceed the pre-defined thresholds, while the yield (egress rate) is relatively low (Fig. 19). It means that the platform is incapable to provide the service correctly and will soon experience a degradation in service delivery.

### 6.3.2 Implementation and results

The implementation is composed of three steps (Fig. 20):

- attack simulation;
- alarms generation by the monitoring engine;
- alerts generation using CRS.

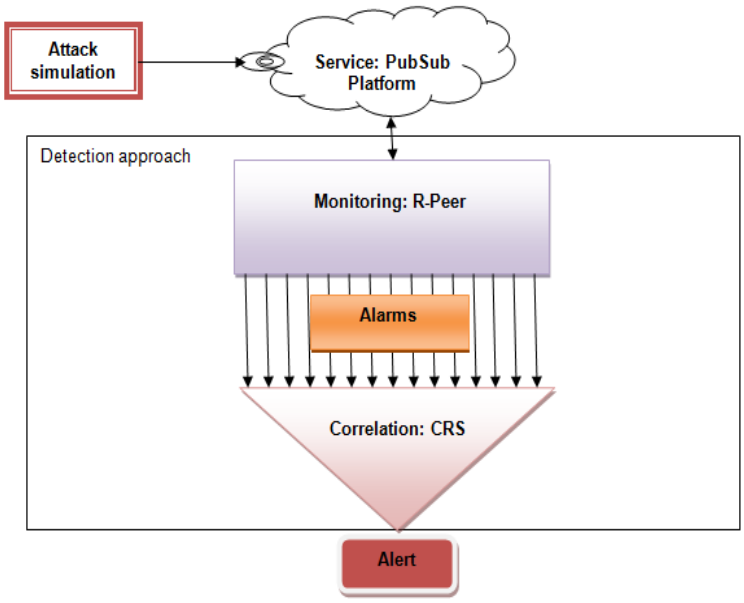



Figure 20: Description of the experimentation

**Simulation**

The first step of the experimentation uses SolAdvisor which allows the following:

- users creation: we have started by creating a closed user group (CUG) with 1000 publishers and 1000 subscribers;
- each publisher is able to send (publish) 100 MB in 50 seconds;
- all publishers connect and start publishing at the same time, making the platform overwhelmed and incapable to treat all their messages.

Fig. 21 shows how publishers are created with SolAdvisor, i.e., "add publisher" button ( ), and how to have them publishing a set of messages, i.e., "start publish" button ( ).

**Alarms generation**

As explained in the previous section, to detect the flooding attack, we supervise the level of:

- connections;

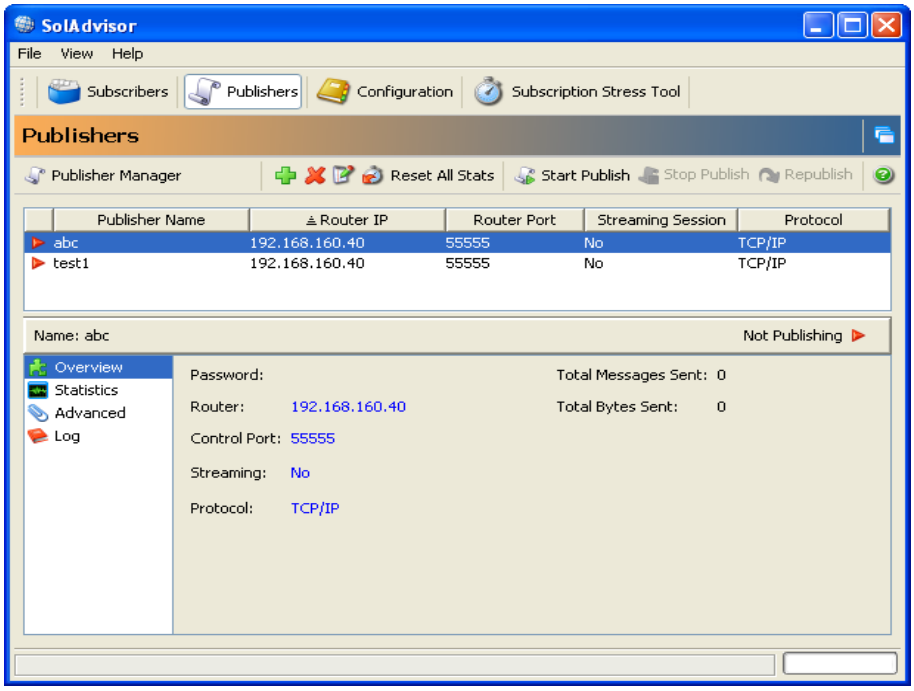



Figure 21: SolAdvisor-publisher manager view

- ingress message rate;
- disk utilization;
- egress message rate.

For each one of these variables, we define two kinds of thresholds, and we configure the SNMP agent to send a trap for warning Nagios each time a threshold is exceeded:

- *set-value*: high trap set threshold value from 1 to 9999 - an event is triggered each time the trap value rises above this threshold;
- *clear-value*: low trap clear threshold value from 1 to 9999 - an event is cleared each time the trap value falls below this threshold.

**Alerts generation**

When traps are received by Nagios, this latter will display them on its graphical interface on one hand. On the other hand, the alarms are stored into a log file and sent to CRS which will correlate them with other events and trigger an alert when a chronicle is recognized (see below).

If a problem occurs, one can see it on the Nagios interface, i.e., a cloud symbol appears in the "Service" column next to the "TRAP" (Fig. 22-(A)). More details about the alarm can also be displayed, e.g., in Fig. 22-(B) where the trap indicates that an alarm concerning the ingress message rate was received.

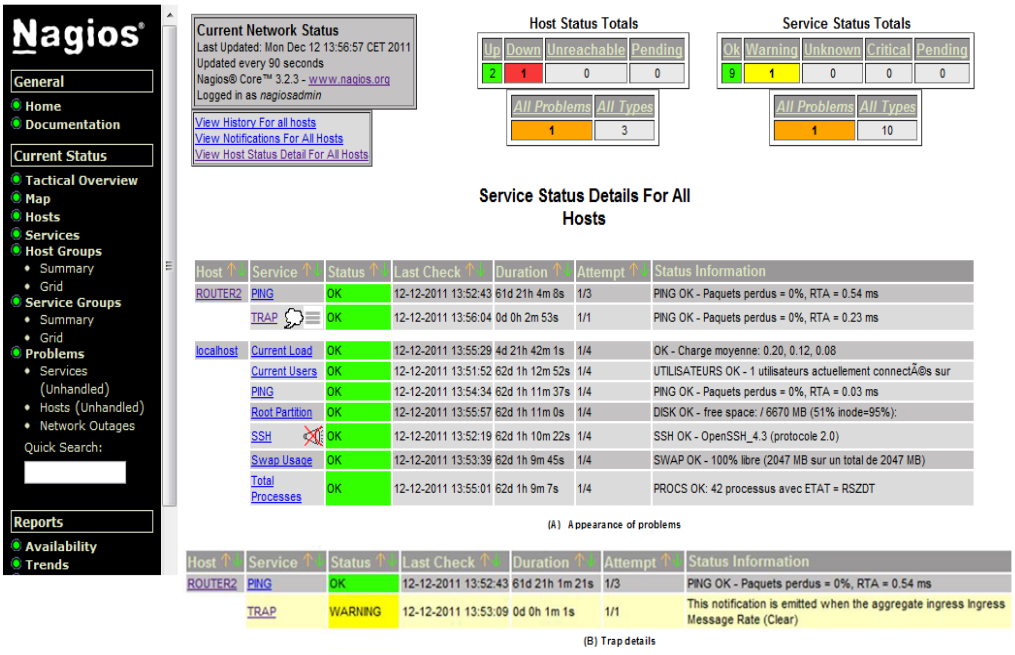



Figure 22: Router 2 supervision using Nagios - Problems appearance

## 6.4 Conclusions and lessons learnt

The main goal of our experimentation was to improve challenge detection mechanisms as part of the ResumeNet $D^2R^2 + DR$ resilience strategy. We have proposed a detection mechanism based on two steps: monitoring, followed by correlation. The monitoring system introduced here is based on a separation between the data plane and the service plane. In the data plane are defined the metrics (disk and bandwidth utilization) we are monitoring, while the service plane is composed of the tools used for monitoring (SNMP). This separation renders our system autonomic and reusable with different kinds of services and challenges. Thus, this system can be customized, according to the service and the desired level of resilience. For correlation, we have used CRS which is based on temporal aspects for correlation. In his turn, CRS shows a separation between a data plane and a service plane. The data plane is represented by the *chronicle base* where chronicles are defined and stored. Therefore, CRS enforces the autonomy and reusability of our detection system. Nevertheless, CRS can be replaced by any other kind of correlation system, from the moment this latter respects reusability.

We have found that the definition of chronicles is the most important and challenging part of this experimentation. More the chronicle is clear and accurate, more the detection of a challenge is guaranteed. This difficulty led us to focus on the definition of the appropriate chronicle for the detection of flooding attacks.

We noted two more interesting things during this experimentation:

- we came to the conclusion that denial of service attacks are easy to launch but it is hard to protect a system against them - the basic problem is that, generally, services and systems assume that the users behave well;

- the same symptoms may be the results of an attack (DoS, DDoS), or of a legal utilization of the service (flash crowd) - at the service level, the symptoms are the same, a huge load that the service can't handle - however, the reaction should not be the same - the distinction between these cases is not obvious.

# 7 General Conclusions

One of the major goals of the experimentation activities proposed in work package 4 (WP4) has been to determine the extent to which the $D^2R^2 + DR$ resilience strategy can be applied in practical network settings. This network resilience strategy should be understood as a set of guidelines for a methodical approach in the development of resilient networks and networked applications. The specific mechanisms and their interconnection can be dependent on the specific network scenario, as shown in the four case studies which have been the subject of our project.

One of the major conclusions is that there are certain scenarios where most of the functionality is provided by only one component of the strategy, whereas in other scenarios two related components have the same function and therefore operate as one logical block. For the former, an example is given in the description of the first experimentation scenario. Nodes try to discover the network topology and traffic matrix, where due to the distributed nature of the system, no systematic misrepresenting can be done by flows. Here the emphasis is on individual nodes, which based on the data gathered from the network construct their own flow dependency graph. The graph is then used to identify nodes to which the local host should not offer forwarding, since reciprocation could not be achieved. There is obviously a game where each player is a flow and the strategy played is the route to be used. Such a one-shot game does not have a separate phase to deal with free-riding (this would represent recovery and remediation). Therefore, this functionality has to be build into the strategy selection component, which is by definition the *defense* phase. In fact, defense is self-enforcing in this scenario, in the sense that hosts that find each other collaborating as a result of the dependency graph calculation would not like to change their strategy. The second scenario provides an interesting insight into the $D^2R^2 + DR$ resilience strategy as well. Since opportunistic networks are an extremely distributed environment, detection is extremely hard to achieve, being limited essentially to a few special cases. One of the consequences is that the separation between challenged and unchallenged networks is not clear anymore, since even the normal mode can be considered as challenged. It is for this reason that recovery and remediation form here the same functional block.

The last two scenarios have investigated either detailed mechanisms to provide resilience for applications (CoSIP and virtualization) or the application of these principles to enterprise systems (the communicating objects platform operated by France Telecom). A study of migration methods in virtual environments has been provided by University of Passau, which provided an experimental evaluation in this sense. This work deals mostly with the recovery and remediation aspects of our strategy. The use of virtualization techniques has been studied further by University of Passau and Technische Universität München. Two example applications have been considered here: *i)* VoIP using CoSIP for the sake of integration of two resilience mechanisms, namely virtualizaton and P2P signaling, *ii)* and web-based video streaming. Finally, France Telecom has focused on developing techniques for detecting threats to enterprise environments, where they have evaluated the use of the existing monitoring tools in conjunction with the Chronicle Recognition System (CRS), developed in-house.

The application of $D^2R^2 + DR$ strategy to multi-level resilience has proved to be more complex from an experimental point of view. Most of the multi-level applications are in the wireless area, where the interactions between the PHY, MAC and network layer can produce sometimes unexpected results, due to the fact that these protocols were developed independent of each other. Theoretically, in order to provide optimality (or resilience) for the network, a set of coupled optimizations has to be solved, which is usually intractable. Our experiments deal however with a number of multi-level aspects. The incentive protocol for wireless mesh networks uses at the application layer information gathered at network layer, the CRS system can process information collected at multiple levels as well and the migration in virtualized environments has been evaluated on different levels too. We believe that a thorough coverage of multi-level resilience aspects is highly scenario-dependent and has to rely on assumptions and parameters which are characteristic to that particular setting.

## References

[BCIP07] Chiara Boldrini, Marco Conti, Iacopo Iacopini, and Andrea Passarella. Hibop: a history based routing protocol for opportunistic networks. In *Proc. IEEE WoWMoM 2007*, 2007.

[BCM+99] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proc. 19th International Conference on Distributed Computing Systems (ICDCS)*, Austin, TX, USA, 31 May - 4 June, 1999.

[BGR11] Fredrik Bjurefors, Per Gunningberg, and Christian Rohner. Haggle testbed: a testbed for opportunistic networks. In *In Proceedings of the 7th Swedish National Computer Networking Workshop*, 2011.

[Bir93] K.P. Birman. The process group approach to reliable distributed computing. *Communications of the ACM, Vol. 36, No 12*, pages 37–103, December 1993.

[CD00] M.-O. Cordier and C. Dousson. Alarm driven monitoring based on chronicles. In *Proc. 4th Symposium on Fault Detection Supervision and Safety for Technical Processes (SAFEPROCESS)*, pages 286–291, Budapest, Hungary, June 2000.

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proc. 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, pages 273–286, 2005.

[DA08] Luca Deri and Richard Andrews. N2N: A Layer Two Peer-to-Peer VPN. In *Autonomous Infrastr., Management and Security*, pages 53–64, 2008.

[DCABM03] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 134–146, New York, NY, USA, 2003. ACM.

[DMC06] S. Deb, M. Medard, and C. Choute. Algebraic gossip: a network coding approach to optimal multiple rumor mongering. *Information Theory, IEEE Transactions on*, 52(6):2486 – 2507, 2006.

[Fes10] Ali Fessi. *Resilient Application Layer Signaling based on Supervised P2P Networks*. PhD thesis, Technische Universitaet Muenchen, December 2010.

[FFCdM11] Andreas Fischer, Ali Fessi, Georg Carle, and Hermann de Meer. Wide-area virtual machine migration as resilience mechanism. In *International Workshop on Network Resilience (WNR) in conjunction with the IEEE Symposium on Reliable Distributed Systems 2011 (SRDS)*, October 2011.

[FFH+11] Ali Fessi, Andreas Fischer, Ralph Holz, , Chidung Lac, Nils Kammenhuber, and Somia Natouri. Resilient service architecture (final). Technical Report FP7-224619, D3.1c, 7th Framework Programme - Future Internet Research and Experimentation (FIRE), Augst 2011.

[FyLBS+06] Alaeddine El Fawal, Jean yves Le Boudec, Kave Salamatian, A. Self, and Limiting Epidemic Service. Self-limiting epidemic forwarding. Technical report, In The First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC2007, 2006.

[GT02] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM ToN*, 10, 2002.

[HCY08] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In *MobiHoc*, pages 241–250, 2008.

[KOK09] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proc. 2nd Int'l Conference on Simulation Tools and Techniques*, 2009.

[KRH+06] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: practical wireless network coding. *SIGCOMM Comput. Commun. Rev.*, 36:243–254, August 2006.

[LDS03] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE MCCR*, 2003.

[NAGa] `http://www.nagios.org/`.

[NAGb] `http://en.wikipedia.org/wiki/Nagios/`.

[NGR09] Erik Nordström, Per Gunningberg, and Christian Rohner. A search-based network architecture for mobile devices. Technical Report 2009-003, Department of Information Technology, Uppsala University, January 2009.

[Ope10] Open Source Portable SIP Stack and Media Stack for Windows and Mac OS X. `http://www.pjsip.org/`, June 2010.

[PGLK10] Gabriel Popa, Eric Gourdin, Franck Legendre, and Merkouris Karaliopoulos. On Maximizing Collaboration in Wireless Mesh Networks Without Monetary Incentives. In *WiOpt'10: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 331–340, Avignon, France, 2010.

[SGC+09] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD data set cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[SPR05] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *ACM SIGCOMM WDTN Workshop*, 2005.

[Sys10] Solace Systems. Solace messaging platform soladvisor user guide. May 2010.

[TIK11] TIKnet wireless testbed. `http://tiknet.ee.ethz.ch/doku.php/`, December 2011.

# Paper A

# On Realising a Strategy for Resilience in Opportunistic Networks

**Marcus Schöller, Paul Smith, Christian Rohner, Merkouris Karaliopoulos, Abdul Jabbar, James P.G. Sterbenz, and David Hutchison**

Future Network and MobileSummit 2010 Conference, Florence, 2010

Because of our increased dependence on communication networks, resilience needs to be a fundamental property of the future Internet. We define resilience as the ability of a network to provide an acceptable level of service in the light of various challenges, such as episodic connectivity or malicious actors. There have been many helpful point solutions to improve resilience in the Internet, yet a systematic approach is necessary to make resilience a first class citizen of the future Internet.
In this paper, we apply our general resilience strategy, called D2R2 + DR, to an opportunistic networking scenario, showing how it can be used to address the challenge of selfish nodes. The strategy describes a real-time control loop to allow dynamic adaptation of the networked system in response to challenges, and an off-line loop that aims to improve the performance of the network via a process of reflection. We briefly describe our approach to quantifying resilience, and its use in our scenario. Initial simulation results indicate the promise of our approach.

*Future Network and MobileSummit 2010 Conference Proceedings*
*Paul Cunningham and Miriam Cunningham (Eds)*
*IIMC International Information Management Corporation, 2010*
*ISBN: 978-1-905824-16-8*

# On Realising a Strategy for Resilience in Opportunistic Networks

Marcus Schöller[1], Paul Smith[2], Christian Rohner[3], Merkouris Karaliopoulos[4]
Abdul Jabbar[5], James P.G. Sterbenz[2,5], and David Hutchison[2]
[1]*NEC Europe, Kurfürsten-Anlage 36, Heidelberg, 69115, Germany*
*Email: marcus.schoeller@neclab.eu*
[2]*Lancaster University, InfoLab21, Lancaster, LA1 4WA, UK*
*Email:* $\{p.smith|jpgs|dh\}$*@comp.lancs.ac.uk*
[3]*Uppsala Universitet, Box 337, 751 05 Uppsala, Sweden*
*Email: christian.rohner@it.uu.se*
[4]*ETH Zürich, Gloriastrasse 35, 8092 Zürich, Switzerland*
*Email: karaliopoulos@tik.ee.ethz.ch*
[5]*The University of Kansas, Lawrence, KS 66045-7612, US*
*Email:* $\{jpgs|jabber\}$*@ittc.ku.edu*

**Abstract:** Because of our increased dependence on communication networks, resilience needs to be a fundamental property of the future Internet. We define resilience as the ability of a network to provide an acceptable level of service in the light of various challenges, such as episodic connectivity or malicious actors. There have been many helpful point solutions to improve resilience in the Internet, yet a systematic approach is necessary to make resilience a first class citizen of the future Internet.
In this paper, we apply our general resilience strategy, called $\mathsf{D^2R^2} + \mathsf{DR}$, to an opportunistic networking scenario, showing how it can be used to address the challenge of selfish nodes. The strategy describes a real-time control loop to allow dynamic adaptation of the networked system in response to challenges, and an off-line loop that aims to improve the performance of the network via a process of reflection. We briefly describe our approach to quantifying resilience, and its use in our scenario. Initial simulation results indicate the promise of our approach.

**Keywords:** Opportunistic networks, resilience, survivability, DTN

## 1. Introduction

Society increasingly depends on networks in general and the Internet in particular for many aspects of our daily lives. Consumers use the Internet to access information, obtain products and services, manage finances, and communicate. Business entities use the Internet to conduct business with their customers and with one another. Nations rely on the Internet to conduct government affairs, deliver services to their citizens, and, to some extent, manage homeland security and conduct military operations. As its reach and scope continue to extend, the Internet increasingly subsumes services previously implemented on separate networks.

With this increasing dependence on the Internet and the integration of services within it, the disruption of networked services may lead to severe consequences. Lives of individuals, the economic viability of businesses and organizations, and the security of nations are directly linked to the resilience, survivability, and dependability of data networks. Unfortunately, the increased sophistication and interdependence of services render the Internet more vulnerable to industrial espionage, information warfare, and

cyber-crime in general. In parallel, its expansion to the mobile wireless domain exposes the network to the challenges of error-prone links and intermittent network connectivity, raising additional concerns with respect to its robustness and scalability.

The Internet community realised those challenges early, and in many cases has responded to them with resounding success. After all, the very first design choice of datagram routing and maximum flexibility in route retrieval was motivated exactly by the need for high network robustness. Mechanisms such as optical ring restoration, for example, have further signified the attempts to strengthen the network operation tolerance to failures. Nevertheless, there is consensus in the community that the majority of the previous, clearly valuable, efforts have largely been done in isolation rather than as part of an overall systematic approach.

In this paper, we outline an approach to network resilience, based on a general resilience strategy being investigated as part of the EU-funded ResumeNet project [1]. We define network resilience as the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation [2]; effectively, resilience can be viewed as a superset of commonly used definitions for survivability, dependability, and fault tolerance. The strategy describes a real-time control loop to allow dynamic adaptation of a networked system in response to challenges, and an off-line control loop that aims to improve the performance of the network (including the real-time loop) via a process of reflection.

We apply this general resilience strategy to an opportunistic networking scenario, and attempt to illustrate its value. In addition to the application of the strategy to opportunistic networks, we have previously demonstrated its general utility when considering problems of resilience in other contexts, for example, to mitigate the effects of flash crowd events [3] or for weather disruption in millimeter wave wireless mesh networks [4]. Here, however, we show how our opportunistic transport protocol implementation fits into the overall strategy, following the two control loops. These two control loops are triggered by the introduction of an additional challenge to the simulation scenario, namely malicious nodes that do not behave according to the transport service specification, by not forwarding data. We have inserted this challenge into our simulation environment together with mechanisms that aim to allow the system to maintain an acceptable level of service. In the evaluation section we assess the impact of such malicious nodes, as well as the success of our potential remedy, including its costs.

This paper is organized as follows. Our general resilience strategy is described in Section 2. This strategy is investigated within the study case on opportunistic networking in Section 3. Thereafter, the experimentation results derived by our opportunistic networking emulator are presented in Section 4. Our conclusions and an outlook to future work close the paper in Section 5.

## 2. A General Strategy for Network Resilience – $\mathsf{D}^2\mathsf{R}^2 + \mathsf{DR}$

Our research is based on a general strategy for multi-level network resilience, called $\mathsf{D}^2\mathsf{R}^2 + \mathsf{DR}$ [2]. The strategy involves two nested control loops. The inner loop is a real-time adaptation control loop consisting of the Defence, Detection, Remediation, and Recovery stages. The outer loop contains the stages of Diagnosis and Refinement. This strategy was developed by the ResiliNets Initiative [5] and is partly based on a

number of previous strategies, including ANSA [6], CMU-CERT [7], and SUMOWIN [8]

It is common practice to protect networked systems by providing defensive measures. With respect to resilience, two lines of *defence* can be drawn. The first line attempts to prevent challenges from affecting the system, e.g. firewalls. A second line of defence aims to limit erroneous behaviour within a service and tries to prevent failures propagating to other services or to the application. However, since it is impossible to forecast all potential challenges and discover every system fault, defensive measures alone are not sufficient to build a resilient network. Therefore, *detection* mechanisms must be put in place to identify deviations from the specified operational service. The (optimal) result of the detection stage is an informed report about a detected challenge. Based on this report the *remediation* stage applies a resilience strategy in order to maintain the desired level of delivered service, despite the adverse operational condition; or at least it will provide a graceful degradation of the affected service. As soon as the challenge ceases, the activated remediation mechanisms should be discontinued in order to free the resources they consume. We call this deactivation *recovery*.

The outer control loop deals with the long-term improvement of the system. The idea is to assess how successful the real-time control loop was in ensuring network resilience through the various stages outlined above. This is done in the *diagnosis* stage. In the *refinement* stage, the system is improved in response to the outcome of diagnosis, and also includes, for example, reporting to the operator about challenges that cannot be mitigated with the installed resilience mechanisms.

The general strategy abstracts many of the complexities of building resilient networked systems, such as the need for a distributed monitoring system. Many of these complexities are specific to the deployment environment, such as the opportunistic networking scenario presented here. The next section presents details of the system enhancements we have developed for our simulator, in which these control loops are implemented, and we then present the simulation results.

## 3. Resilience for Opportunistic Networks

We now describe the application of our resilience strategy to an opportunistic networking scenario. Access to the world's networks has become a commodity in a large number of countries, where infrastructure, such as optical fibres, is readily available. However, there are vast regions, often remote, sparsely populated, and with a relatively poor economic base, where the deployment of constant connectivity is not a viable option. Projects like N4C [9] or ZebraNet [10] aim to provide basic e-mail and (cached) Web access to such remote areas using opportunistic networks.

### *3.1 Service Specification of a Store-Carry-Forward Transport*

In opportunistic networks, typically mobile nodes store, carry, and forward messages when they encounter other nodes, using short- range communication. A store-carry-forward (SCF) transport service [8] allows the flow of data in the network despite the absence of end-to-end paths. Data instead travels over *space-time paths*, comprised of sets of links that become available in different time instants in the network. Node mobility is thus important for data dissemination; it creates contact opportunities between different nodes and allows nodes physically to transport data to areas where no connectivity might be available.

A misbehaving node may break this service specification by not forwarding data when a space-time path exists. It is important to note that this service specification assumes unlimited storage capacity. The results presented below are also based on this assumption. We are currently working on incorporating node buffer limitations in the simulator to reflect a more realistic opportunistic network. However, this also introduces complications, which we detail in Section 5. pointing to our future work.

### *3.2 Realising the $D^2R^2 + DR$ Strategy*

Based on the resilience strategy described in Section 2., we now illustrate how the opportunistic network can be enhanced to cope with misbehaving nodes. An evaluation of this implementation is provided in the subsequent section. Our overall strategy for mitigating misbehaving nodes is depicted in Figure 1. In summary, we start by understanding the potential capabilities of the network, e.g. in terms of delivery ratio, delay, and number of replicas related to various proposed data forwarding schemes. We do this with the help of simulations and analytical modelling. If we detect a deviation from the expected behavior because of misbehaving nodes, we remediate by adapting the configuration of the store-carry-forward (SCF) transport mechanism. A more detailed description of the realisation of the strategy is as follows.

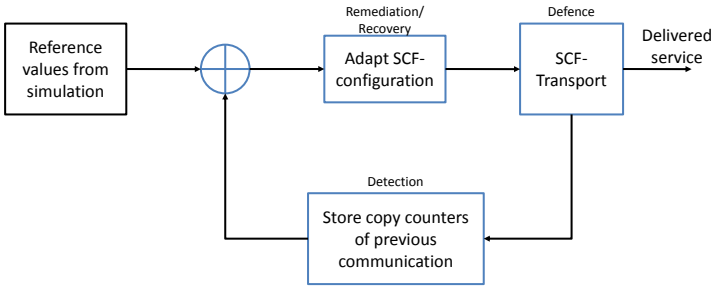



Figure 1: Control loop steering the transport protocol

**Defence** A network for opportunistic communication using a SCF transport service has inherent defence against the challenge of episodic connectivity. The mobility of nodes gives rise to a diversity of space-time paths. Depending on how aggressive a data forwarding scheme is, the network can exploit all or part of it, at the expense of resource consumption. Epidemic forwarding, for example, makes use of all possible space-time paths in the network and achieves minimum message transfer delay; yet, it generates a very high number of message replicas in the network. Two-hop forwarding, on the other hand, exploits only part of this diversity, limiting the length of feasible space-time paths to two hops [11]. This same diversity can be the counter-measure against misbehaving nodes, whether selfish or malicious, which fail to contribute to the transport service. Likewise, the network defence against selfish behaviours could be strengthened by implementing game-theoretic mechanisms with or without money to promote or enforce node cooperation. An example of a mechanism without money is given in [12]; a similar approach is being developed in the ResumeNet project.

**Detection** The implementation of challenge detection mechanisms is one of the most difficult parts in an opportunistic network, precisely because of their intermittent

connectivity. To detect the presence of misbehaving (or unhelpful) nodes, sources would need to maintain a history of the nodes that participated in the delivery of a message. In turn, receiver nodes should maintain a list of nodes they have seen and nodes that successfully delivered a message to them. Then the sender and receiver nodes would need to exchange their state, either upon a direct contact or during an off-line period, e.g., when the devices are attached to some infrastructure. Using that information, the sender could deduce which nodes cooperated in forwarding a given message, and which did not; and out of them, which nodes did so because they never saw the receiver, and which ones chose not to transmit the message.

Extensions of this algorithm include nodes sharing their local knowledge about the utility of various nodes – by disseminating information about unhelpful nodes and those that are useful for certain destinations. The utility of this algorithm in different forwarding scenarios and the effect of various parameters of the algorithm on its utility (e.g. the amount of state to maintain) require further investigation.

**Remediation** If a node detects the presence of misbehaving nodes, it adapts the SCF transport service configuration to enable a more aggressive forwarding mechanism. For example, it could shift from two-hop to epidemic forwarding. Effectively, in this way, it restores some or all of the space-time diversity that is lost due to the existence of misbehaving nodes.

**Recovery** The use of epidemic forwarding as a remedy against misbehaving nodes brings the associated cost of increased energy consumption and buffer utilisation at the network nodes (see Section 4. for details). Therefore, the SCF transport service should recover to its normal operation using two-hop forwarding as soon as the malicious nodes have disappeared from the system. This requires additional detection capabilities, which are currently under further investigation.

**Diagnosis** The diagnosis step includes the understanding of the impact of node misbehaviours on the network performance and the ability of the remediation solution to cope with it. For example, when the remediation is implemented via use of the epidemic forwarding scheme, all data can be forwarded to its destination and the delivery delay is decreased. However, the diagnosis also reveals the high costs that this forwarding scheme places on the system.

**Refinement** Based on the diagnosis step, measures for better balancing between the resource usage of the scheme and the achievable end-user performance can be designed. An ageing mechanism for efficiently managing the node forwarding storage is such an example. Messages older than a certain threshold will be deleted from the store. The results of this measure are also shown below.

### *3.3 Resilience Metrics for Opportunistic Networking*

One of the difficult tasks is the quantitative characterisation of resilience in order to evaluate the *efficacy* of architectures and developed mechanisms. This is an especially hard problem because of the numerous 'levels' within networks and the interaction between these levels. Given our multi-level resilience approach, we are developing a framework that enables resilience evaluation at any arbitrary level. First, we define a service at any given layer boundary. We then quantify the resilience of the network at

this boundary using a two-dimensional state-space model [13]. Along one dimension, we characterise the *service* at a given layer boundary using the metrics that are desired from such a service (e.g. storage size). Along the other dimension, metrics that define the *operational* state at the layer boundary (i.e. metrics that affect those defined in the service dimension) are specified (e.g. data delivery ratio). Finally, we quantify resilience as a measure of service degradation in the presence of challenges (perturbations) to the operational state of the network.

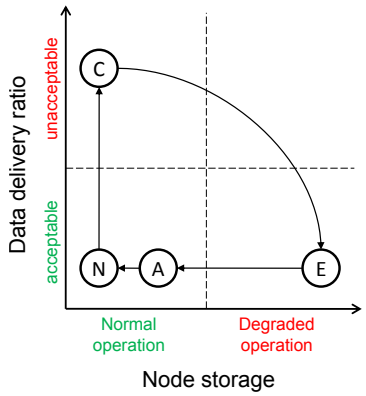



Figure 2: Resilience metric state space

In Figure 2, this approach is depicted for our opportunistic networking scenario.

The system's normal operation (N) is affected by the presence of misbehaving nodes, which degrade the provided service (C). Applying epidemic forwarding as a remedy allows the system to deliver the desired service again, but at an increased cost (E). Introducing ageing during the refinement phase improves the system to maintain the specified service, while reducing the cost again (A). Finally, recovery brings the system back into its normal mode of operation (N).

## 4. Evaluation

We sketch an opportunistic network scenario to evaluate the proposed resilience strategy. The experimental results are obtained with the Haggle experimentation architecture [14] running on 20 (virtual) nodes with controlled connectivity. Connectivity between two nodes follows a two state Markov model with typical average contact time and average inter-contact time of 30 seconds and 150 seconds respectively. This topology avoids large connected clusters but still gives enough contact opportunities to exchange data. The nodes generate data every 2 seconds with a random destination among the 20 nodes. Our emulator offers the possibility of dynamically changing the forwarding strategy and resource management policies, for example to age out data carried for a long time. We use the two-hop forwarding scheme [11] with no data ageing as the default configuration (as normal operation).

Our experiment considers four operational states: *normal* operation with all nodes using the two-hop forwarding strategy, *challenged* operation with 8 out of the 20 nodes refusing to forward data of other nodes, as an expression of selfish behaviour. Assuming perfect detection, we pass into the, third, *remediation* phase by activating epidemic forwarding on the remaining co-operating nodes. Finally, we enter the *refinement* phase by periodically ageing data based on the time they stay in a node's data buffer; thus, we

compensate for the additional redundancy introduced by epidemic forwarding. For our evaluation we consider the amount of data received at the destination, the end-to-end delay for all received data, and the amount of data in the buffer of a co-operating node, measured over a period of 90 seconds. The results are summarized in Table 1.

| scenario | delivered data | end-to-end delay | buffer |
|---|---|---|---|
| normal (N) | 227 | 37 s | 127 |
| challenged (C) | 134 ($-41\%$) | 31 s ($-16\%$) | n/a |
| remediation (E) | 226 ($\pm 0\%$) | 18 s ($-51\%$) | 942 ($+642\%$) |
| refinement (A) | 246 ($+8\%$) | 17 s ($-54\%$) | 177 ($+39\%$) |

Table 1: Evaluation results.

Selfish behaviour of 8 out of the 20 nodes reduces the number of delivered data significantly to 59% of the data delivered during normal operation. The end-to-end delay of delivered data is reduced by a few seconds, because data with a longer delivery time in normal operation is not delivered at all in challenged operation and thus does not contribute to the result. By using epidemic forwarding as a remediation mechanism, the co-operating nodes compensate for the impact of selfish nodes, achieving almost the same delivery of data as in normal operation. Furthermore, end-to-end delay of the delivered data is much smaller because the restriction for redundancy of two-hop forwarding no longer applies, but data may reach the destination over longer space-time paths. However, because redundancy is no longer restricted, the amount of data on the inspected node increases dramatically from 127 units during normal operation to 942 during the remediation phase. The refinement process takes account of that aspect by ageing data that was stored on a node for more than 45 seconds. As a result, the stored data is reduced to 177 units, which is again close to the normal operation. End-to-end delay was not affected, while the number of delivered messages actually improves.

This latter aspect is explained as follows: our emulation software ranks - in order of importance – the data to be transferred to another node, and transfers only some of the data to help limit congestion. Also, the longer data is in the buffer of a node, the higher the likelihood that it already has been delivered to the destination by other nodes. By ageing older data, we thus give priority to the transfer of newer data.

## 5. Conclusion

In this paper, we have argued that resilience is a necessary building block for any future network. Despite the fact that many resilience mechanisms have been added to networks, especially to the Internet, a systematic approach to resilience has not so far been developed in order to increase its availability and survivability. We have introduced a general strategy that aims to embed resilience systematically into networked systems. We have applied our strategy to an opportunistic networking scenario, showing some of our early results and how this strategy can enhance the network over time.

Future work will refine the service specification for the store-carry-forward transport service to reflect realistic resource limitations of the network nodes. Due to this limitation several complications arise; these impact the detection mechanisms, i.e. how to detect misbehaving nodes, in contrast to nodes which dropped data as a result of limited storage. Moreover, epidemic forwarding as a remediation mechanism can worsen the system performance compared to normal two-hop forwarding in the presence of

some misbehaving nodes. Repeatedly applying our strategy to the simulator should enable us to find suitable mechanisms and more importantly validate more completely the suitability of our resilience strategy.

## Acknowledgements

The research was supported in part by the European Commission under grant FP7-224619 (ResumeNet) and US NSF FIND under grant CNS-0626918. The authors would like to thank members of these projects for their input into the ideas in this paper.

## References

[1] "The EU FP7 Resumenet Project." `http://www.resumenet.eu`.

[2] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, 2010. (to appear).

[3] L. Xie, P. Smith, D. Hutchison, M. Banfield, H. Leopold, A. Jabbar, and J. P. G. Sterbenz, "From detection to remediation: A self-organized system for addressing flash crowd problems," in *IEEE ICC 2008*, pp. 5809–5814.

[4] A. Jabbar, J. P. Rohrer, A. Oberthaler, E. K. Çetinkaya, V. S. Frost, and J. P. G. Sterbenz, "Performance comparison of weather disruption-tolerant cross-layer routing algorithms," in *IEEE INFOCOM 2009*, pp. 1143–1151.

[5] "The ResiliNets Initiative." `https://wiki.ittc.ku.edu/resilinets`.

[6] N. Edwards, "Building dependable distributed systems," Technical report APM.1144.00.02, ANSA, February 1994.

[7] R. J. Ellison *et al.*, "Survivable network systems: An emerging discipline," Tech. Rep. CMU/SEI-97-TR-013, PA, 1999.

[8] J. P. G. Sterbenz *et al.*, "Survivable mobile wireless networks: issues, challenges, and research directions," in *ACM WiSE 2002*, pp. 31–40.

[9] "The N4C project." `http://www.n4c.eu`.

[10] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002.

[11] M. Grossglauser and D. N. Tse, "Mobility increases the capacity of ad hoc wireless networks," in *IEEE/ACM Trans. on Networking*, vol. 10, pp. 477–486, August 2002.

[12] L. Buttyán, L. Dóra, M. Félegyházi, and I. Vajda, "Barter trade improves message delivery in opportunistic networks," *Ad Hoc Net.*, vol. 8, no. 1, pp. 1–14, 2010.

[13] A. J. Mohammad, D. Hutchison, and J. P. Sterbenz, "Poster: Towards quantifying metrics for resilient and survivable networks," in *IEEE ICNP 2006*, pp. 17–18.

[14] E. Nordström, P. Gunningberg, and C. Rohner, "A search-based network architecture for mobile devices," Tech. Rep. 2009-003, Uppsala Univ., 2009.

# Paper B

# Trace-based performance analysis of opportunistic forwarding under imperfect cooperation conditions

**Merkouris Karaliopoulos and Christian Rohner**

INFOCOM 2012 mini-conference, Orlando, 2012

Abstract: The paper proposes an innovative method for the performance analysis of opportunistic forward- ing protocols over files logging mobile node encounters (contact traces). The method is modular and evolves in three main processing steps. It first carries out trace inflation to systematically identify those contacts that constitute message forwarding opportunities for given message coordinates and forwarding rules. It then draws on graph expansion techniques to capture these forwarding contacts into sparse space-time graph repre- sentations (constructs). Finally, it runs standard short- est path algorithms over these constructs and derives typical performance metrics such as message delivery delay and path hopcount. The method is flexible in that it can easily assess the protocol operation under various expressions of imperfect node cooperation. We describe it in detail, analyze its complexity, and assess it against discrete event simulations for three repre- sentative randomized forwarding schemes. The match with the simulation results is excellent and obtained with run times up to three orders of size smaller than the duration of the simulations, thus rendering our method a valuable tool for the performance analysis of opportunistic forwarding schemes.

.

# Trace-based performance analysis of opportunistic forwarding under imperfect cooperation conditions

Merkourios Karaliopoulos
Department of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece
Email: mkaralio@di.uoa.gr

Christian Rohner
Department of Information Technology
Uppsala University
Uppsala, Sweden
Email: christian.rohner@it.uu.se

***Abstract*—The paper proposes an innovative method for the performance analysis of opportunistic forwarding protocols over files logging mobile node encounters (*contact traces*). The method is modular and evolves in three main processing steps. It first carries out *trace inflation* to systematically identify those contacts that constitute message forwarding opportunities for given message coordinates and forwarding rules. It then draws on *graph expansion* techniques to capture these *forwarding contacts* into sparse space-time graph representations (constructs). Finally, it runs standard shortest path algorithms over these constructs and derives typical performance metrics such as message delivery delay and path hopcount. The method is flexible in that it can easily assess the protocol operation under various expressions of imperfect node cooperation. We describe it in detail, analyze its complexity, and assess it against discrete event simulations for three representative randomized forwarding schemes. The match with the simulation results is excellent and obtained with run times up to three orders of size smaller than the duration of the simulations, thus rendering our method a valuable tool for the performance analysis of opportunistic forwarding schemes.**

## I. Introduction

The performance analysis of forwarding protocols in opportunistic networks has been carried out with various means. Analytical models, largely drawing on Markovian chains and Ordinary Differential Equations, have proven extremely powerful in computing performance metrics such as message delivery probability, delay and hopcount [4] [7] [21]. However, all forwarding schemes do not lend themselves to analytical treatment; and even for those that do, the aforementioned analytical models are supported by certain assumptions about the node mobility patterns (*i.e.*, exponential pairwise inter-contact times) that do not hold but only under few random mobility models [14].

Real traces of mobile node encounters, a.k.a contact traces, are viewed as more realistic representations of node mobility and, therefore, have found enormous acceptance in the opportunistic networking community. broadly speaking, traces have been used in two ways in opportunistic networking studies. In several studies, traces have been themselves the subject of analysis in an attempt to extract insights to fundamental properties of the nodes' mobility such as contact durations and intercontact times [2], the number and diversity of space-time paths generated by this mobility [19]. In even more studies, traces have been interfaced with discrete-event simulators, such as ONE [9], to support protocol performance evaluation tasks. However, the practical value of this use, suffers from scalability issues in that run times grow fast even for moderate number of messages and trace sizes.

Our paper effectively combines these two main uses of traces. We are interested in the performance analysis of opportunistic forwarding schemes, both when nodes fully cooperate in the forwarding process and when they behave selfishly, (selectively) deferring from it. But instead of feeding the traces to some simulator, we work directly with them and extract these metrics through their analysis. Our method proceeds in three phases. Firstly, the contact traces are inflated and filtered. The trace inflation process, which is described in detail in Section III, replicates contact records in a way that eases the sequential parsing of the contact trace. The filtering of contact records is carried out per message and forwarding protocol and retains those contacts that represent real message forwarding opportunities in line with the protocol rules. In a second step drawing on graph expansion ideas, the retained contact records are converted to a space-time graph construct; this is effectively a tree with the message source node at its root. Finally, standard shortest path algorithms are run over this construct to yield the space-time path resulting in minimum message delivery and derive performance metrics such as message probability delivery and path hop count.

Our scripts modulate appropriately the contact-filtering step to accommodate different forwarding schemes under various expressions of node selfish behavior. The resulting analysis is approximative in that it does not account for the impact of network resource contention in a way that a discrete event simulator can do. Nevertheless, these approximations stand in excellent match with the results obtained from the most popular opportunistic networking simulator across the research community. Most importantly, the run times of the trace-based analysis are two and three orders of size smaller and let experimentation

scale with the number of messages and the size of traces. Regarding the trace size, in particular, the trace inflation step introduces modest flexibility for trading the accuracy of the analysis with its run time. Therefore, we argue that our analysis can become a valuable tool for researchers that work on the performance analysis of opportunistic forwarding schemes.

We review related work on representing and analyzing dynamic network topologies in Section II. Our trace-based analysis is introduced in detail in Section III under the assumption that nodes fully cooperate in the message forwarding process; it is expanded to scenarios of imperfect cooperation in Section IV; and is assessed with respect to its accuracy and run times with the help of discrete event simulations in Section V. We conclude with some thoughts on the value of our analysis in Section VI.

## II. Related Work

Dynamic network topologies, whereby the relative positions of nodes and links between them change over time, have been largely analyzed through appropriate adaptations of standard static graph attributes. We can distinguish two main approaches in doing this; each results in different graph constructs $G_c = (V_c, E_c)$ and facilitates different algorithms for their analysis.

In the first approach, the set of graph vertices $V_c$ coincides with the set of network nodes $V$ and edges are defined for all node pairs *i.e.*, $E_c = E(K_{|V|})$. The dynamics of the network are captured into time-varying edge weight(cost) functions $w_e(t)$, expressing the cost(delay) of traversing link $e$ over time. This cost has, at least, two components: the waiting time $q_e(t)$ till a message the link becomes available and the actual transmission time over the link $p_e(t)$. What changes across different studies are the assumptions about these two functions. For example, Jain *et al.* in [6] explicitly assume that the functions $p_e(t)$, hence also the weight functions $w_e(t)$, respect the FIFO property: two messages traverse a link in the order they are transmitted over it. Chaintreau *et al.* in [3] and Bui Xuan *et al.* in [20] make more implicit yet stricter assumptions about $p_e(t)$. In both cases, $p_e(t)$ is piecewise constant, taking either zero [3] or finite [20] values for the duration of a contact between two nodes and resulting in discontinuous piecewise linear functions $w_e(t)$, as shown in Fig. (1). Standard shortest path (SP) algorithms can be then applied to find the fastest routes that messages can follow across the network, after replacing the typically constant edge weights $w_e$ with the time-varying weight functions $w_e(t)$. Much earlier in [16], and independently from the research thread on opportunistic networking, Orda and Rom had derived algorithms for minimum-delay paths in networks with continuously time-varying link delays. They relax the FIFO assumption for $w_e(t)$ at the expense of slightly higher complexity for the resulting variant of the Dijkstra SP algorithm.

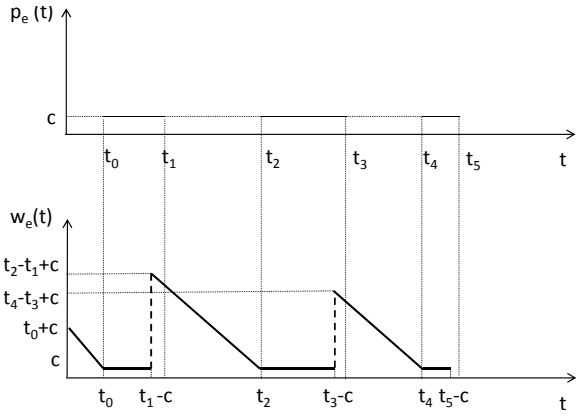



Fig. 1: Constant transmission time, $p_e(t)$, and corresponding link traversal delay function, $w_e(t)$, for the dynamic network link model in [3] and [20].

Whereas, with the first approach, the standard SP algorithms are adapted for time-varying edge weight functions, the second approach adapts the original graph attributes via *graph-expansion* techniques to let the SP algorithms run without modifications. Merugu *et al.* [13] and Erramilli *et al.* in [19] use space-time graphs. They define discrete time epochs and construct for each one of them a different graph snapshot with full replication of the original node set $V$. On the contrary, Kostakos in [10] introduces a more scalable space-time graph representation by replicating vertices and edges only upon occurrence of a link(contact) between them. As in [13] and [19], he explicitly discriminates between edges joining identical nodes across successive time epochs and edges connecting different nodes upon a given time epoch. He uses this construct to analyze both unidirectional (email communication) and bidirectional (phone conversations) communication patterns and computes different metrics based on minimum delay paths.

The graph construct we use draws on graph-expansion techniques and, in particular, the construct in [10]. In Section III we describe how the construct can be adapted to account for the peculiarities of opportunistic encounters and the rules of different forwarding schemes. Later in Section IV we further expand it to account for non-cooperative node behavior. In contrast with other work so far in literature, our graphs capture neither the social (contact) graph nor the space-time graph summarizing the full sequence of node encounters; *they are rather message-specific forwarding contact trees capturing those encounters that actually contribute to the forwarding process under certain protocol rules.*

## III. Shortest paths under perfect cooperation

The computation of shortest space-time path(s) for a given message evolves in three phases. The phases differentiate according to the forwarding scheme we consider and whether shortest paths correspond to minimum message delivery delay or minimum path hopcount. In every case,

input to the first processing phase are traces of node encounters, *i.e.*, sequences of contact records with the general format shown in Fig. 3(a). Each contact record $C_k = (n1, n2, t_{s,k}, t_{e,k})$ includes four fields: the two nodes that encounter each other, the time the encounter starts, $t_s(c_k)$, and the time it ends, $t_e(c_k)$. The contact records are stored in increasing order of their start times.

In the remainder of this section, we present the computation steps under the assumption that the duration of each contact suffices for the transfer of a message from one node to another(*zero link transmission delay* assumption, $c$=0 in Fig. 1). Hence, messages can be described by the triple $m = (s, d, t_g)$, where $s(m)$ and $d(m)$ are the message source and destination nodes, respectively, and $t_g(m)$ the time the message is generated or first becomes available at some opportunistic network node. In [8], we discuss how we can relax the zero link transmission delay assumption to account for finite positive sizes of messages and link transmission rates.

### *A. From the original contact trace to the sequence of forwarding contacts*

*1) Inflating the original trace:* The original trace is first *artificially inflated*. Each contact record $C_k = (n1, n2, t_{s,k}, t_{e,k})$ is replicated as many times, $r(c_k)$, as the number of contact records that start later than $C_k$ but before its end time

$$r(C_k) = |\mathcal{C}_{on}(C_k)|,\ \mathcal{C}_{on}(C_k) = \{C_j | t_{s,k} \leq t_{s,j} \leq t_{e,k}\} \quad (1)$$

Each of the $r(C_k)$ contact records is a replica of $C_k$ except for its start time field, which is set to the start time of the respective contact in $\mathcal{C}_{on}(C_k)$, *i.e.*, $C_{k,j} = (n1, n2, t_{s,j}, t_{e,k})$, $0 \leq j \leq r(C_k) - 1$. The trace inflation process is carried out once per trace. Each parsed contact enters a list $L$ storing ongoing contacts (lines 8 and 22 of the algorithm's pseudocode). As subsequent contacts are parsed, those contacts in $L$ that are found to have finished are discarded (line 16), whereas those that are still ongoing are replicated after adapting the start time field (line 19). An example of the trace inflation process is given in Fig. 2.

*Why trace inflation?* There are two high-level requirements from our technique. It has to be computationally friendly and scalable while uncovering all forwarding opportunities. The sequential parsing of the contact records according to their start time serves the first objective; however, it does not account correctly for the duration of encounters. For example, in Fig. 2, a message generated at node 6 at time 710 can reach node 1 instantaneously through the path 6-5-1, induced by the time-overlapping contacts $C_3$ and $C_4$. To capture this, the parser should be continuously backtracing contact records, comparing the start times of currently parsed records against the end times of earlier records. But this increases substantially the computational overhead of parsing.

**Algorithm 1** Trace inflation process

```
1: INPUT     original tracefile D
2: OUTPUT    inflated trace D_inf
3: VARS      L : list storing ongoing contact records
4:
5: INITIALIZATION
6: copy 1^st contact record C_1 from D to D_inf
7: add C_1 to L
8:
9: while !eof(D) do
10:       read C_k from D;
11:       write C_k in D_inf;
12:       foreach contact C_j in L
13:         if t_e(C_j) < t_s(C_k)
14:             remove C_j from L
15:         else
16:             write C_j in D_inf, t_s(C_j) = t_s(C_k)
17:         end
18:       end
19:       add C_k to L
20: end while
```

The artificial trace inflation, is effectively a compromise between the two requirements. Forwarding opportunities become visible to the parser at the expense of additional parsing effort for otherwise irrelevant contact records. However, the process cannot alleviate misses completely. In the same example of Fig. 2, a message generated at the node 6 any time $t$ in $[702, 800]$ can reach the node 3 over the (space-) path presented by the time-overlapping contacts $C_0$, $C_3$, and $C_4$. Yet, the order of the replicated contact records in the inflated trace does not let the sequential parser anticipate this.

The inflation overhead increases with the duration of encounters and the extent of their overlap in time. It is lower for sparse traces of occasionally meeting nodes and increases fast when nodes meet frequently with each other. The impact of the inflation process on both the accuracy and speed of our technique is assessed in Section V.

*2) Filtering for forwarding contacts:* This step, hereafter called *contact-filtering* step, is closely dependent on the forwarding scheme under analysis each time. The inflated trace is time-sorted according to the contact start time field and filtered with criteria accounting for the particular forwarding rules. The aim of the contact-filtering step is to retain only those contacts that can result in data forwarding, hereafter called *forwarding contacts*. If $t_g$ is the time a message $m$ becomes available at source node $s$ for destination node $d$, then the contact-filtering step first excludes all contact records up to either (a) the earliest still ongoing contact $c_b$ involving the message source node, with $t_s(c_b) < t_g \leq t_e(c_b)$; or, when no such ongoing contacts exist, (b) the first contact $c_a$ involving node $s$ after time $t_g$. It then initializes the *forwarding list* with the node $s$. The forwarding list stores each time *potential* message forwarding nodes, *i.e.*, nodes that have acquired the message and may, depending on whom they encounter, forward it further.

Subsequent contact records are parsed sequentially and

| contact id | involved nodes | | contact start time | contact end time |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| $C_0$ | 1 | 3 | 589 | 940 |
| $C_1$ | 2 | 3 | 589 | 619 |
| $C_2$ | 6 | 7 | 639 | 699 |
| $C_3$ | 1 | 5 | 700 | 816 |
| $C_4$ | 5 | 6 | 702 | 818 |
| $C_5$ | 2 | 7 | 816 | 1185 |
| $C_6$ | 4 | 8 | 819 | 909 |
| $C_7$ | 8 | 6 | 938 | 1182 |
| ... | ... | ... | ... | ... |

| contact id | involved nodes | | contact start time | contact end time |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| $C_{0,0}$ | 1 | 3 | 589 | 940 |
| $C_1$ | 2 | 3 | 589 | 619 |
| $C_{0,1}$ | 1 | 3 | 589 | 940 |
| $C_2$ | 6 | 7 | 639 | 699 |
| $C_{0,2}$ | 1 | 3 | 639 | 940 |
| $C_{3,0}$ | 1 | 5 | 700 | 816 |
| $C_{0,3}$ | 1 | 3 | 700 | 940 |
| $C_{4,0}$ | 5 | 6 | 702 | 818 |
| $C_{0,4}$ | 1 | 3 | 702 | 940 |
| $C_{3,1}$ | 1 | 5 | 702 | 816 |
| $C_{5,0}$ | 2 | 7 | 816 | 1185 |
| $C_{0,5}$ | 1 | 3 | 816 | 940 |
| $C_{3,2}$ | 1 | 5 | 816 | 816 |
| $C_{4,1}$ | 5 | 6 | 816 | 818 |
| $C_6$ | 4 | 8 | 819 | 909 |
| $C_{0,5}$ | 1 | 3 | 819 | 940 |
| $C_{5,1}$ | 2 | 7 | 819 | 1185 |
| $C_7$ | 8 | 6 | 938 | 1182 |
| $C_{0,6}$ | 1 | 3 | 938 | 940 |
| $C_{5,2}$ | 2 | 7 | 938 | 1185 |
| ... | ... | ... | ... | ... |

Fig. 2: Original and inflated trace with 8 and 20 contact records, respectively.

may belong to one of three typologies, depending on the encountered nodes:

- neither node lies in the forwarding list (*0-entry contacts*).
- both nodes are already listed in the forwarding list (*2-entry contacts*).
- one of the two nodes is in the forwarding list (*1-entry contacts*).

Contacts of the first type do not contribute to the forwarding process and are left out. Likewise, we filter out contacts of the second type. They do not represent real forwarding opportunities since the common assumption in all schemes we analyze is that nodes with a message copy will forward it to a node that does not have it and is eligible to acquire it upon the first encounter with it. On the contrary, 1-entry contacts represent the most interesting contact type and their manipulation depends directly on the opportunistic forwarding scheme. Below, we explicate the manipulation of 1-entry contacts for three representative schemes coming under the broader family of controlled flooding (randomized forwarding): the epidemic [21], two-hop [4] [5], and source spray-and-wait [18] protocols.

*1-entry contacts under the epidemic scheme*: Under epidemic message routing, nodes that avail the message forward it to all nodes that do not. Hence, all 1-entry contacts are also forwarding contacts and are retained.

*1-entry contacts under the two-hop scheme*: Under two-hop message routing, nodes other than the message source availing a message copy cannot forward it but only to the destination node. Therefore, two types of 1-entry contacts are retained: those $(s, *, t), t \geq t_s$ involving the source node $s$ as the single node already logged down in the forwarding list, as well as the first 1-entry contact involving the destination node. All other 1-entry contacts are filtered out of the trace.

*1-entry contacts under the source spray-and-wait scheme*: The scheme is a generalization of two-hop routing, in that the source node can copy a message to the first $F$ different nodes it encounters. Then, the two-hop variant we described corresponds to $F = |V| - 1$. As with the two-hop scheme, the first 1-entry contact involving the destination is the last one to include in the sequence of the forwarding contacts. However, only the first $F$ 1-entry contacts involving the source node are retained and the length of the forwarding list is upper bounded by $F + 1$ rather than $|V| + 1$, which is the case for the two-hop scheme.

For example, in Fig. 3(a), contacts $C_0$ and $C_7$ are forwarding contacts for a message $m = (n1, n4, t_g),\ t_g < t_{0,s}$ for all three schemes; whereas contact $C_3$ is a forwarding contact only for the epidemic scheme.

*Implementation complexity:* From an implementation point of view, finding the first contact record $(s, *, t')$, $t' \geq t_s$, requires a binary search with argument $t_s$ within the full contact trace. If $N$ is the total number of time-ordered contact records after the trace inflation step, the time complexity of this search is $O(log_2 N)$. We then need to read all contact records *after* $(s, *, t')$, which takes $O(N)$ time. The parsing of each record involves two searches and potentially (when parsing 1-entry contacts) an insertion of an element in the forwarding list. The list can be implemented with a binary heap using the node index as key. It involves $O(|V|)$ insert operations, where $V$ is the set of network nodes, each costing $O(log_2|V|)$ time, and $O(2N)$ search operations at a cost of $O(log_2|V|)$ each. Therefore, the overall complexity of step 1 is $O(Nlog_2(|V|) + N + log_2 N + |V|log_2|V|) = O(Nlog_2|V|)$, which, for typical cardinality values of the node set $V$, is in the order of the time needed to read the contact records.

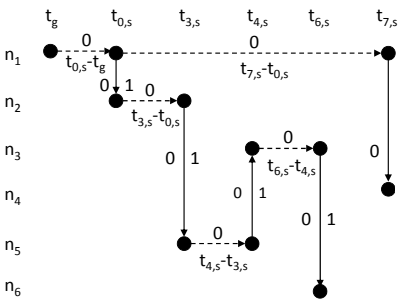
### B. Building the forwarding contact graph

The set of forwarding contacts out of the first processing phase make up the shortest space-time paths towards the message destination node $d$ *and all other nodes that can be reached earlier than* $d$. The next step is to derive the graph construct $G_c = (V_c, E_c)$ that can model these contacts and their timing relationship. The construct iterates the "temporal" graph representation in [10].

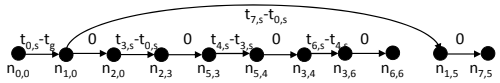
For all forwarding schemes in question (epidemic, two-hop, source spray-and-wait), the graph construct is initialized by a vertex corresponding to the message source node and gets updated by 1-entry forwarding contacts occurring thereafter, adding to the graph: a) a pair of vertices, one for each node involved in the contact; b) one *space-spanning* directed edge connecting the two encountered nodes; and c) one *time-spanning* directed edge towards

| contact id | involved nodes | | contact start time | contact end time | additional fields |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| $C_0$ | n1 | n2 | $t_{0,s}$ | $t_{0,e}$ | ... |
| $C_1$ | n3 | n4 | $t_{1,s}$ | $t_{1,e}$ | ... |
| $C_2$ | n4 | n5 | $t_{2,s}$ | $t_{2,e}$ | ... |
| $C_3$ | n2 | n5 | $t_{3,s}$ | $t_{3,e}$ | ... |
| $C_4$ | n5 | n3 | $t_{4,s}$ | $t_{4,e}$ | ... |
| $C_5$ | n3 | n2 | $t_{5,s}$ | $t_{5,e}$ | ... |
| $C_6$ | n3 | n6 | $t_{6,s}$ | $t_{6,e}$ | ... |
| $C_7$ | n1 | n4 | $t_{7,s}$ | $t_{7,e}$ | ... |
| $C_8$ | n2 | n6 | $t_{8,s}$ | $t_{8,e}$ | ... |
| ... | ... | ... | ... | ... | ... |

(a) Forwarding contacts




(b) Graph construct.




(c) Linearized DAG graph.

Fig. 3: Forwarding contacts (in bold), graph construct $G_c$, and linearized DAG for message $m = (n1, n4, t_g), t_g < t_{0,s}$ under epidemic forwarding. Solid(dash) lines correspond to space(time)-spanning edges; $n_{i,j}$ denotes the $G_c$ vertex corresponding to node $n_i$ at time $j$.

the node that is already included in the forwarding list, originating from the vertex that represents its most recent forwarding contact. Hence, every time a node $v \in V$ appears in an 1-entry forwarding contact, it generates a new vertex $v_c \in V_c$ for construct $G_c$. There are two exceptions to this. Firstly, for forwarding contacts $C_j$ that are *ongoing* by the time the message becomes available ($t_s(C_j) < t_g(m) < t_e(C_j)$). Such contacts do not introduce time-spanning edges. Secondly, for forwarding contacts, whereby the same nodes simultaneously encounter more than one nodes without a message copy. These contacts generate only space-spanning edges as well, one for each simultaneous encounter. In the end of this process, each network node $v \in V$ is eventually identified with a single global index in $[1, |V|]$ for the node set $V$ and the forwarding list, and multiple non-successive indices for the construct vertex set $V_c$.

The graph edge set $E_c$ is weighted. When we are interested in minimum message delivery delay, time-spanning edge weights equal the time lag between successive appearances of the same node in forwarding contacts and express the time over which a message may be stored and carried by a given node. Space-spanning edges express the time it takes to forward a message upon a contact and, under the zero link transmission delay assumption, are assigned zero weights. On the contrary, when we are interested in minimum hopcount, time(space)-spanning edges are assigned zero(unit) weights, as shown in Fig. 3(b).

*Implementation complexity.* To construct the graph $G_c$, we need to store for each node $v \in V$ a) the most recent time instant it gets involved in a forwarding contact; and b) the respective vertex index assigned to it in $G_c$. If nodes are placed in an ordered array of size $|V|$, both reading and updating these values take $O(1)$ time. An additional linked list stores for each node $v \in V$ all vertex-indices that are assigned to them each time they are involved in a forwarding contact. This is necessary so that the nodes involved in the shortest space-time path can be backtraced once the shortest path is computed over the graph construct $G_c$ (see section III-C).

### C. Computing message shortest paths over the forwarding contact graph

The last processing step consists in the computation of shortest space-time paths over the expanded graph $G_c$. We can state the following proposition.

*Proposition 3.1:* For all three opportunistic forwarding schemes, the complexity of finding shortest paths in $G_c$ is $O(V)$.

*Proof:* The parsing of contacts ends upon the first appearance of the message destination node $d$ in a forwarding 1-entry contact. This may happen while the forwarding list length lies anywhere in $[1, N_f]$, where $N_f = |V|$ for epidemic and two-hop forwarding and $N_f = F + 2 \leq |V|$ for source spray-and-wait forwarding, after the processing of $N_f - 1$ forwarding 1-entry contacts.

For all three controlled flooding schemes, the expanded graph $G_c = (V_c, E_c)$ features vertex cardinality $|V_c| = 2N_f + 1 = O(2|V| - 1) = O(|V|)$ and edge cardinality $|E_c| = 2N_f = O(2|V| - 2) = O(|V|)$. Therefore, $G_c$ is a sparse directed acyclic graph (DAG). We can first linearize it through running a Depth-First-Search (DFS) over it (see [8] for an example) and then directly run the Dijkstra shortest path algorithm to get the shortest space-time paths (Fig. 3(c)). Both algorithmic steps, as well as the overall processing step feature time complexity $O(|V_c| + |E_c|) = O(|V|)$. ■

It is quite straightforward to extend these modular processing steps to relax the zero link transmission delay assumption and account for multi-node encounters and multiparty end-to-end communication patterns. We refer the reader to [8] for a detailed analysis of these extensions.

## IV. Minimum-delay paths under imperfect cooperation

In Section III, we could derive the performance of different forwarding schemes under the assumption that all nodes cooperate in the forwarding process. In reality, this is rarely the case. The scarcity of device resources (energy and battery), lack of trust, and privacy concerns may motivate less cooperative node behavior. The ways

comm1=[1 3 5 7 8], comm2=[2 4 5 6], comm3=[1 2 4 7]

| contact id | involved nodes | | contact start time | contact end time | additional fields |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| **$C_0$** | **n1** | **n2** | **$t_{0,s}$** | **$t_{0,e}$** | **...** |
| $C_1$ | n3 | n4 | $t_{1,s}$ | $t_{1,e}$ | ... |
| **$C_2$** | **n4** | **n2** | **$t_{2,s}$** | **$t_{2,e}$** | **...** |
| **$C_3$** | **n1** | **n5** | **$t_{3,s}$** | **$t_{3,e}$** | **...** |
| $C_4$ | n2 | n8 | $t_{4,s}$ | $t_{4,e}$ | ... |
| **$C_5$** | **n7** | **n4** | **$t_{5,s}$** | **$t_{5,e}$** | **...** |
| $C_6$ | n3 | n6 | $t_{6,s}$ | $t_{6,e}$ | ... |
| **$C_7$** | **n6** | **n2** | **$t_{7,s}$** | **$t_{7,e}$** | **...** |
| $C_8$ | n7 | n4 | $t_{8,s}$ | $t_{8,e}$ | ... |
| ... | ... | ... | ... | ... | ... |

Fig. 4: Forwarding contacts (entries in bold) for the message $m = (n1, n6, t_g), t_g < t_{0,s}$ in the presence of social selfishness (3 communities). The otherwise forwarding contacts $C_4$ and $C_6$ are removed in the contact-filtering step.

in which a node may modulate its forwarding behavior vary broadly. In the remainder of this section, we discuss two representative expressions of non-cooperative behavior and describe how the three-phase process described in Section III –its contact-filtering step, in particular– can be adapted to yield the performance of the same forwarding schemes. Further expression of such behaviors and their analysis can be found in [8].

### A. Encounter-agnostic selfishness

Nodes may well adopt a uniform policy towards messages originating from other network nodes, irrespective of the message source node or the nodes they encounter and ask them to relay some message [7]. In other words, they make no discrimination towards the other network nodes but rather treat all in the same way. We could identify two variants of such encounter-agnostic selfishness:

*1) Fully defer from relaying others' messages:* Nodes unwilling to relay any message that does not originate from themselves. This policy may be triggered due to very tight energy constraints in devices with little autonomy or complete lack of trust when a device operates in an unknown environment. The device is effectively a *black hole* node for others' messages so that none of the encounters involving it correspond to forwarding opportunities.

The treatment of black hole nodes during the trace parsing is rather straightforward and can be assisted by a *black list* storing the selfish nodes. For messages that do not originate from and are not destined for black hole nodes, all contacts involving one or more such node(s) can be filtered out of the original trace before the per-message parsing starts. Then the three-phase parsing can proceed over the residual smaller-size trace without additional online checks against the black list. The cost of this trace-level filtering is effectively amortized over all messages not involving black hole nodes as their sources or destinations. Under a uniform selection of message source and destination nodes, the expected number of these messages is $M_s = M \cdot (1 - n_{bh})^2$, where $n_{bh}$ is the percentage of black hole nodes in the network and $M$ the number of processed messages. For the remaining $M - M_s$ messages involving black hole nodes as their source and/or destination any filtering is carried out at the message level and cannot result in additional time savings.

### B. Encounter-aware selfishness

Contrary to encounter-agnostic selfishness, nodes may vary their forwarding policy towards different network nodes. Encounter-aware selfishness could capture the more usual case, where a node moves in an environment with a mix of familiar/trusted and unknown nodes. Such forwarding policy differentiation could also emerge at the level of social communities, in cases that the network exhibits strong structure: nodes forward messages only from/to nodes belonging to the same community. This is sometimes called "social selfishness" in the literature [11] [12]. The communities may be discrete, forming a network partition, or overlapping, whereby some nodes are members of more than one community, as in Fig. 4.

In the most general and computationally-demanding case, each network node is associated with a set of friend nodes, whose messages it is willing to forward. This is the case for the relay-oriented social selfishness under overlapping asymmetric communities, where a list of friend nodes has to be associated with each single network node. Upon each 1-entry contact $(nx, ny, t)$, the filtering of the trace for forwarding contacts needs to consult the friendship list of the node that is already logged in the forwarding list. Therefore, we need one additional search over an $O(|V|)$ list requiring $O(log_2|V|)$ time for each parsed entry.

## V. Evaluation

### A. Methodology and traces

Our evaluation uses three well-known experimental traces of pairwise node encounters. The traces, listed in Table I, have been gathered over the last five years in the context of the Haggle Project [2] and report Bluetooth sightings by groups of users carrying iMotes during the experiments. Each Bluetooth sighting is assumed to be a contact whereby nodes can exchange information. The logged contacts include both encounters between iMotes ('internal' contacts) and encounters with other Bluetooth-enabled devices ('external' contacts) in the vicinity of iMote carriers. In this study, we focus on internal contacts: these represent data transfer opportunities among the explicit participants of the experiment, assuming that they are all equipped with always-on devices. The experimental settings are detailed in [2].

For each trace, we generate message triplets $msg(s, d, t_s)$, where the message source $s$, destination $d$, and generation time $t_s$ are randomly chosen. Following the processing described in Sections III and IV, we emulate the path presented to the message according to the rules of each forwarding scheme. We implement three scenarios for each combination of trace and forwarding

TABLE I: Characteristics of experimentation datasets

| Configuration | Intel | Cambridge | Infocom05 |
|---|---|---|---|
| Device type | iMote | iMote | iMote |
| Duration(days) | 6 | 6 | 4 |
| Scan time(sec) | 5-10 | 5-10 | 5-10 |
| Granularity(sec) | 120 | 120 | 120 |
| Mobile Devices | 8 | 12 | 41 |
| Stationary Dev. | 1 | 0 | 0 |
| External Dev. | 119 | 211 | 233 |
| Average internal contacts/pair/day | 9.09 | 12.09 | 8.60 |
| # of Contacts | 2766 | 6732 | 28216 |

scheme. The first one assumes that all nodes cooperate fully in the forwarding process (*perfect cooperation*). The second involves $N_1/2$, $N_2/3$, and $N_3/3$ black hole nodes for the three traces, respectively, where $N_i$ is the number of iMote nodes in trace $i$ (*black hole selfishness*). Finally, a third scenario introduces four overlapping communities of socially selfish nodes (*social selfishness*). The community sizes range from 1/3 to 1/2 of the nodes appearing in the trace. Runs with 1000 and 10000 messages are carried out.

The code has been implemented in MATLAB®, release 2010b, using the rather modest support of the Bioinformatics toolbox for graph-theoretic analysis. The same traces and message sequences are then fed to an independent simulator, the ONE [9], to drive a thorough discrete-event simulation. We chose ONE because it has been written explicitly for the simulation of opportunistic protocols and is used by the majority of the DTN community, rather than any other optimality reason. We compare our method against the ONE simulations with respect to:

*a) run times.* The run times of ONE simulations are measured on a 2.7GHz Intel Core i7 machine with 8GB of memory, whereas the trace-based evaluation is carried out on a 2.3 Intel Core i3 laptop with 4GB memory. Hence, the comparison is slightly pessimistic for the trace-based evaluation. Nevertheless, this comparison is not meant to be pedantic; it rather attempts to give a rough idea of the computational savings that are achievable by our method.

*b) accuracy.* We compute standard performance metrics of interest to someone carrying out performance analysis: the ratio of delivered messages, $P_D$, the average message delivery delay, $E[D]$, and hopcount, $E[h]$, as reported by the two performance analysis alternatives. The results of the discrete-event simulations serve as the "ground truth" against which our results are assessed.

### B. Results

*1) Run times:* Table II compares the run times of the trace-based analysis (trace) with the duration of the discrete-event simulations (ONE) under perfect node cooperation. For $10^4$ messages we have only carried out trace-based analysis since the durations of ONE simulations are prohibitive. The run-time advantage of the trace-based evaluation is higher for smaller trace sizes, its execution

TABLE II: Run times of ONE- and trace-based (both for the original and inflated traces) analysis (in secs): perfect cooperation.

| | | Intel | | Cambridge | | Infocom05 | |
|---|---|---|---|---|---|---|---|
| | msgs | $10^3$ | $10^4$ | $10^3$ | $10^4$ | $10^3$ | $10^4$ |
| epid | infl | 9 | 86 | 23 | 188 | 712 | 7394 |
| | orig | 4 | 48 | 6 | 52 | 28 | 285 |
| | ONE | 9656 | | 12459 | | 81077 | |
| 2hop | infl | 8 | 82 | 27 | 229 | 916 | 8003 |
| | orig | 4 | 40 | 6 | 56 | 30 | 268 |
| | ONE | 5963 | | 6944 | | 41019 | |
| ssaw | infl | 9 | 85 | 29 | 244 | 1022 | 8893 |
| | orig | 4 | 40 | 6 | 57 | 29 | 257 |
| | ONE | 2732 | | 3392 | | 2987 | |

times being even three orders of size smaller than the time required by ONE to do the same work. For example, according to Table II, it takes our scripts less than 12 mins to compute message delivery times for $10^3$ messages for the Intel trace under the perfect cooperation scenario, whereas it takes approximately a day to do the same with ONE.

The run times for the same analysis under imperfect cooperation do not have a uniform trend for all forwarding schemes. For the black hole selfishness scenario, they get worse for epidemic forwarding but stay similar or even decrease for two-hop and spray-and-wait forwarding. Table III also shows that epidemic forwarding becomes computationally more demanding than the other two schemes. On the contrary, the social selfishness scenario deteriorates the run-times of all three forwarding schemes since it involves more checks against the community lists and introduces higher parsing overhead. ONE simulations, on the other hand, get lighter in the presence of imperfect cooperation since fewer events are generated in the course of simulation runs. Epidemic forwarding generates the most events, hence, the longer simulation runs.

The growth of run times with the trace size is super-linear for our scripts. The processing of 1000 messages requires only a few seconds for the Intel trace, in the range of half to a couple of mins for the Cambridge trace, and from 8 up to 50 mins for the longer Infocom trace. In comparison, ONE requires up to three orders of size more time for all traces, with different forwarding schemes placing highly variable computational requirements.

*2) Accuracy of trace-based analysis:* We experimented with different combinations of $\{msg\ size, link\ Tx\ rate\}$ in ONE: $\{1bit, 1Gbps\}$ (to match the zero link transmission delay assumption), $\{1kB, 100Mbps\}$, $\{1kB, 10Mbps\}$, and $\{1k, 2Mbps\}$. In all cases, ONE computed identical values for all three performance metrics.

In Table IV, the predictions of the trace-based analysis are in excellent match with the simulator's results –with the exception of average message transmission delays for the first trace under imperfect cooperation, which are 5-15% overestimated by our method. The figures reported in Tables IV-V for the performance of the three schemes are inline with expectations. The epidemic forwarding

TABLE III: Run times (in secs) of trace- and ONE-based analysis: imperfect cooperation.

| | | black hole selfishness | | | | | | social selfishness | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Intel | | Cambridge | | Infocom05 | | Intel | | Cambridge | | Infocom05 | |
| | msgs | $10^3$ | $10^4$ | $10^3$ | $10^4$ | $10^3$ | $10^4$ | $10^3$ | $10^4$ | $10^3$ | $10^4$ | $10^3$ | $10^4$ |
| epid | trace | 14 | 216 | 43 | 585 | 1027 | 10853 | 21 | 212 | 100 | 969 | 883 | 8487 |
| | ONE | 3109 | | 12021 | | 53168 | | 4848 | | 9615 | | 53190 | |
| 2hop | trace | 8 | 104 | 30 | 360 | 545 | 5250 | 29 | 286 | 130 | 1253 | 1269 | 12436 |
| | ONE | 2699 | | 5978 | | 23615 | | 2717 | | 4435 | | 18508 | |
| ssaw | trace | 8 | 105 | 30 | 368 | 585 | 5615 | 29 | 287 | 129 | 1262 | 1390 | 12879 |
| | ONE | 2366 | | 4886 | | 8625 | | 1885 | | 2652 | | 2834 | |

TABLE IV: Trace- *vs.* ONE-based analysis: perfect cooperation, $10^3$ messages.

| | | Intel | | | Cambridge | | | Infocom05 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | metric | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ |
| epid | trace | 996 | 54899 | 2.29 | 637 | 42481 | 2.81 | 890 | 24859 | 4.36 |
| | ONE | 994 | 54841 | 2.23 | 636 | 42711 | 2.62 | 890 | 24888 | 4.18 |
| 2hop | trace | 994 | 57995 | 1.73 | 611 | 47973 | 1.73 | 890 | 25200 | 1.9 |
| | ONE | 993 | 58085 | 1.71 | 609 | 48296 | 1.72 | 881 | 25214 | 1.89 |
| ssaw | trace | 994 | 58276 | 1.73 | 607 | 47526 | 1.73 | 880 | 31110 | 1.83 |
| | ONE | 988 | 58638 | 1.71 | 602 | 48170 | 1.71 | 878 | 31486 | 1.79 |

TABLE VI: Trace-based analysis for the original and inflated traces: perfect cooperation, $10^3$ messages.

| | | Intel | | | Cambridge | | | Infocom05 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | metric | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ |
| epid | infl | 996 | 54899 | 2.29 | 637 | 42480 | 2.81 | 890 | 24859 | 4.3 |
| | orig | 995 | 55354 | 2.22 | 636 | 42648 | 2.5 | 890 | 25021 | 3.9 |
| 2hop | infl | 994 | 57994 | 1.73 | 611 | 47973 | 1.73 | 890 | 28310 | 1.9 |
| | orig | 993 | 59196 | 1.70 | 610 | 48107 | 1.71 | 889 | 28367 | 1.87 |
| ssaw | infl | 994 | 58276 | 1.73 | 607 | 47526 | 1.73 | 880 | 31110 | 1.83 |
| | orig | 993 | 59477 | 1.70 | 606 | 47659 | 1.70 | 879 | 31155 | 1.80 |

delivers consistently the highest number of messages at the minimum time at the expense of more aggressive network resource (battery, airtime) utilization. For the Intel and Cambridge datasets, the performance of two-hop forwarding coincides with the source spray-and-wait. In the existence of black hole nodes and community structure, the superior relaying flexibility of two-hop forwarding is canceled out and both schemes end up running with the same set of nodes as potential relays. For the Infocom dataset, two-hop forwarding gains a small advantage suggesting that nodes mix adequately to enable practically the same number of one- or two-hop space-time paths, even when message replication is restricted in the way source spray-and-wait does this.

*3) Impact of the trace-inflation process:* Table II also highlights the impact of the trace inflation process on the run time of our scripts. The run times of our method for the inflated traces are approximately 2, 5, and 30 times higher, respectively, when compared to those for the original traces. On the other hand, Table VI suggests that the approximations of the trace-based analysis for the performance metrics, when based on the original traces, are quite close to those based on the inflated traces. Except for a couple of exceptions related to hopcount values, most of the figures are less than 1% apart. Therefore, our method provides some flexibility to trade accuracy with run times and computational complexity.

### C. Implications-discussion

The favorable results of the comparisons in the earlier sections do not intend to promote our method over discrete event simulators for the performance analysis of opportunistic forwarding schemes. Each analysis methodology has certain drawbacks and limitations. Computing shortest paths over traces, in particular, cannot account for resource contention at the node buffers or over the air. In this sense, our method yields performance *approximations* that become tighter at conditions of low traffic contention. ONE, and discrete event simulators in general, are invaluable for reproducing realistically a richer variety of scenarios and capturing the impact of various parameters.

On a similar note, the matching of our approximations with the actual protocol performance, depends on the actual application that runs over the opportunistic networks. For instance, our method will give better predictions for the delays experienced by the messages of a mobile microblogging application, *i.e.*, an adaptation of Twitter for opportunistic environments [1], where messages are small and sent quite intermittently by different nodes. On the contrary, it will behave worse for applications generating larger message(file) sizes, such as multimedia content dissemination.

Finally, another way to see our technique is as a way to compute shortest paths accounting for both the mobility patterns of nodes *and* the forwarding protocol rules. Therefore, it can be used for the fast computation of *forwarding scheme-aware* variants of centrality metrics, such as betweenness centrality or its destination aware variant [17], over dynamic network topologies.

## VI. Conclusions

Our paper proposes an innovative systematic method for efficiently approximating the performance of opportunistic forwarding protocols over real traces of mobile node encounters (contacts). At the core of our method, which bypasses discrete event simulations, lie trace-inflation and graph-expansion techniques. Through their combined use, space-time graph constructs are derived out of the contact sequences for each message and forwarding scheme. Standard shortest path algorithms can then be efficiently

TABLE V: Trace- *vs.* ONE-based analysis: imperfect cooperation, $10^3$ messages.

| | | black hole selfishness | | | | | | | | | social selfishness | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Intel | | | Cambridge | | | Infocom05 | | | Intel | | | Cambridge | | | Infocom05 | | |
| | metric | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ | $P_D$ | $D_{avg}$ | $h_{avg}$ |
| epid | trace | 976 | 71474 | 1.66 | 619 | 44066 | 2.42 | 884 | 28101 | 3.82 | 750 | 82506 | 2.13 | 615 | 69802 | 2.64 | 888 | 30438 | 4.32 |
| | ONE | 975 | 62874 | 1.67 | 619 | 43373 | 2.43 | 887 | 27081 | 3.6 | 750 | 78270 | 2.15 | 613 | 68210 | 2.65 | 889 | 29073 | 4.35 |
| 2hop | trace | 976 | 71579 | 1.6 | 604 | 49751 | 1.71 | 883 | 30900 | 1.85 | 626 | 82333 | 1.59 | 427 | 56154 | 1.61 | 848 | 34443 | 1.88 |
| | ONE | 975 | 60810 | 1.61 | 604 | 49013 | 1.7 | 885 | 29867 | 1.85 | 631 | 78772 | 1.59 | 425 | 54980 | 1.59 | 849 | 33703 | 1.87 |
| ssaw | trace | 976 | 71579 | 1.6 | 602 | 49444 | 1.71 | 876 | 32930 | 1.79 | 626 | 82558 | 1.59 | 423 | 55219 | 1.6 | 810 | 36253 | 1.83 |
| | ONE | 975 | 60741 | 1.62 | 602 | 48703 | 1.7 | 881 | 32006 | 1.80 | 626 | 82558 | 1.59 | 423 | 55219 | 1.6 | 810 | 36253 | 1.83 |

applied over these small sparse graphs to extract the space-time paths messages follow under the control of different forwarding schemes.

The method can compute the number of delivered messages, message delivery delay, and the shortest path hopcount. It is approximative in that it cannot account for resource contention within the network. Nevertheless, the match with the results of the most popular simulator for opportunistic networking is excellent and is achieved at computation run times up to three orders of size smaller than the simulator. Skipping the trace inflation process does help the run times scale better for big traces; it is then up to the users of the method to opt for this at the expense of some accuracy penalty, which was found small in our experiments. The computational savings pertain across different evaluation scenarios featuring perfect node cooperation and different expressions of selfish node behavior thus rendering the method a promising tool for the performance evaluation of opportunistic networking protocols.

In this paper we have applied the method to opportunistic forwarding schemes exercising some form of controlled message flooding. We are currently exploring ways to apply it to the analysis of protocols that explicitly assess the relaying utility of encountered nodes, including recently proposed socioaware forwarding protocols. Our first steps in this direction are reported in [15].

## Acknowledgment

This work is partially supported by the European Union Marie Curie grant Retune (FP7-PEOPLE-2009-IEF-255409) and the ResumeNet collaborative research project (FP7-224619).

## References

[1] S. Allen, G. Colombo, and R. Whitaker. Uttering: Social microblogging without the internet. In *Proc. 2nd ACM/SIGMOBILE Int'l Workshop on Mobile Opportunistic Networking (MobiOpp)*, pages 58–64, Feb. 2010.

[2] A. Chaintreau et al. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. IEEE INFOCOM*, pages 1–13, Apr. 2006.

[3] A. Chaintreau, A. Mtibaa, L. Massoulie, and C. Diot. The diameter of opportunistic mobile networks. In *Proc. ACM CoNEXT*, pages 1–12, 2007.

[4] R. Groenvelt. *Stochastic models in mobile ad hoc networks.* University of Nice, Sophia Antipolis, France, Apr. 2005.

[5] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Netw.*, 10:477–486, Aug. 2002.

[6] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.*, 34:145–158, Aug. 2004.

[7] M. Karaliopoulos. Assessing the vulnerability of dtn data relaying schemes to node selfishness. *IEEE Commun. Lett.*, 13(12):923–925, December 2009.

[8] M. Karaliopoulos and C. Rochner. Trace-based performance analysis of opportunistic forwarding. Technical report, online: http://www.csg.ethz.ch/people/karaliom/trace-analysis.pdf, July 2011.

[9] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proc. 2nd Int'l Conference on Simulation Tools and Techniques*, 2009.

[10] V. Kostakos. Temporal graphs. *Physica A-statistical Mechanics and Its Applications*, 388:1007–1023, 2009.

[11] Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *Proc. IEEE INFOCOM*, pages 1–9, Mar. 2010.

[12] Y. Li, P. Hui, D. Jin, L. Su, and L. Zeng. Evaluating the impact of social selfishness on the epidemic routing in delay tolerant networks. *IEEE Commun. Lett.*, 14(11):1026–1028, Nov. 2010.

[13] S. Merugu, M. Ammar, and E. Zegura. Routing in space and time in networks with predictable mobility. Technical report, School of Computer Science, Georgia Tech, Atlanta, USA, 2004.

[14] P. Nain, D. Towsley, B. Liu, and Z. Liu. Properties of random direction models. In *Proc. IEEE INFOCOM*, pages 1897 – 1907 vol. 3, Mar. 2005.

[15] P. Nikolopoulos, T. Papadimitriou, P. Pantazopoulos, M. Karaliopoulos, and I. Stavrakakis. How much off-center are centrality metrics for routing in opportunistic networks. In *ACM MobiCom 2011 CHANTS Workshop)*, Las Vegas, NV, USA, Sept. 2011 (to appear).

[16] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of ACM*, 37:607–625, Jul. 1990.

[17] P. Pantazopoulos, M. Karaliopoulos, and I. Stavrakakis. Centrality-driven scalable service migration. In *23rd International Teletraffic Congress (ITC 2011)*, San Francisco, CA, USA, Sept. 2011 (to appear).

[18] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected networks: the multi-copy case. *IEEE/ACM Trans. on Netw.*, 16(1):77–90, Feb. 2008.

[19] M. C. V. Erramilli, A. Chaintreau and C. Diot. Diversity of forwarding paths in pocket switched networks. In *Proc. 7th ACM SIGCOMM IMC*, pages 161–174, 2007.

[20] B. B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Int'l Journal of Foundations of Comp. Science*, 14:267–285, Jul. 2003.

[21] X. Zang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. *Comp. Networks*, 51(10):2867–2891, Jul. 2007.

# Paper C

# On Resilience in Opportunistic Networks

**Christian Rohner, Fredrik Bjurefors, Maria Mehrparvar, Paul Smith**

Technical Report

We use the ResumeNet resilience metric framework to study the resilience of opportunistic networks. We apply challenges affecting both node contacts and node behavior. Resilience is studied in different network topologies using dif- ferent forwarding protocols.
We find that the diversity in space-time paths give a high degree of resilience to contact and node failures, offering alternative paths to deliver data. The performance decrease is typically graceful until a significant amount of the node contacts or nodes are affected, depending on the diversity offered by the topology. The behavior is similar for most of the studied forwarding protocols, with the exception of the Spray and Wait routing protocol which surprisingly even showed cases where the data delivery delay improved under challenges affecting node contacts.

# On Resilience in Opportunistic Networks

Christian Rohner, Fredrik Bjurefors,
Maria Mehrparvar
Uppsala University, Sweden

Paul Smith
Lancaster University, UK

## ABSTRACT

We use the ResumeNet resilience metric framework to study the resilience of opportunistic networks. We apply challenges affecting both node contacts and node behavior. Resilience is studied in different network topologies using different forwarding protocols.

We find that the diversity in space-time paths give a high degree of resilience to contact and node failures, offering alternative paths to deliver data. The performance decrease is typically graceful until a significant amount of the node contacts or nodes are affected, depending on the diversity offered by the topology. The behavior is similar for most of the studied forwarding protocols, with the exception of the Spray and Wait routing protocol which surprisingly even showed cases where the data delivery delay improved under challenges affecting node contacts.

## 1. INTRODUCTION

In opportunistic networks, typically mobile nodes store, carry, and forward messages when they encounter other nodes, using short-range communication. A store-carry-forward transport service allows a flow of data in the network despite the absence of end-to-end connectivity. Data instead travels over space-time paths, comprised of sets of links that become available in different time instants in the network. Node mobility is thus important for data dissemination; it creates contact opportunities between different nodes and allows nodes to physically transport data to areas where no connectivity might be available.

The store-carry-forward transport offers an inherent resilience against the challenge of episodic connectivity. The connectivity (i.e., space-time paths) itself offers diversity, while replication of the data introduces both spatial and temporal redundancy. Routing protocols have been proposed that have different performance characteristics, i.e., buffer usage, delivery ratio, or delay. However, it is not clear how well these routing protocols perform when subjected to challenges other than episodic connectivity. Knowledge of this could be used to make an informed protocol selection or improve protocol design.

In this paper we evaluate the resilience of different routing protocols to challenges in the context of a number of connectivity patterns. We do this using the ResumeNet metric framework that can be used to evaluate the behavior of opportunistic networks under challenge.

## 2. PRELIMINARIES

### 2.1 Resilience preliminaries

We define resilience as the ability of a network to provide an acceptable level of service in the light of various challenges, such as episodic connectivity, resource limitations, or malicious actors. In previous work [1] we have applied the ResumeNet resilience strategy, called $D^2R^2 + DR$, to an opportunistic networking scenario, showing how it can be used to address the challenge of selfish nodes. The strategy describes a real-time control loop to allow dynamic adaptation of the networked system in response to challenges, and an off-line loop that aims to improve the performance of the network via a process of reflection.

In this work, we study the impact of challenges to better understand their impact on the service level of opportunistic networks. The results might be used to enable detection and identification of challenges.

### 2.2 Routing Protocols

Routing protocols for opportunistic networks try to find a balance between fast and reliable delivery on one hand, and resource use on the other. Using all space-time paths from a source to a destination would introduce too many message replicas filling up buffers on the nodes. Furthermore, it is not known a priori which path that is the best. There are two typical approaches to the problem: limiting epidemic routing by reducing the number of message replicas, or making informed decisions based on contact history or statistics [2, 3, 4].

In this paper we consider one routing protocol of both approaches: *Source Spray and Wait* [5] that limits the number of message replicas to the first few nodes encountered by the source node, and *Prophet* [2] that maintains a probability to deliver messages for a specific destination. Messages are only passed to nodes that have a higher probability than oneself. As a benchmark, we compare to *Epidemic* routing that

makes use of all space-time paths at the cost of high resource usage.

## 2.3 Challenge Description

Challenges that affect the usage of a node contact to transfer data between nodes due to limited abilities (e.g., interference, full buffers, low battery), node failure, active attacks by third parties, or because of selfish node behavior. We abstract from the concrete challenge by selectively removing *node contacts* from contact traces, or excluding *nodes* from participating in the data forwarding process. We do the selection both randomly and informed:

- Removing node contacts: we remove random or spatially correlated node contacts. The spatial correlation is implemented by considering a *jammer* within which range we remove node contacts.
- Selfish node behavior: we exclude random or strategically selected nodes from participating in the data forwarding process. The strategic selection of nodes takes node centrality and betweenness centrality into account, thus excluding nodes that are likely to contribute most to the forwarding process under normal operation.

# 3. RESILIENCE METRICS

The ResumeNet metric framework is a general framework for assessing the resilience of a system. The resilience of a network is assessed as the degree of the network's capability to withstand challenges during a given time interval. A challenge is a series of $n$ elementary changes to which the sequence $R[k]_{0\leq k\leq n}$ can be associated. This definition thus implies that a challenge can be a complex challenge consisting of a well-defined number $n$ of elementary changes. In general, however, $n$ can be a random variable and is not necessarily a-priori known.

Figure 1 sketches three realizations of a challenge of a network consisting of n elementary changes: the middle curve is a random realization, while the upper and lower curves reflect the maximum and minimum of (possibly several) extreme realizations.

The extreme value over the entire interval may not be a realization of the perturbation process, but an upperbound, thus the maximum of extreme realizations. The region between the maximum sequence, $R_{max}[1]$ , $R_{max}[2]$ , ... , $R_{max}[n]$, and the minimum sequence, $R_{min}[1]$ , $R_{min}[2]$ , ... , $R_{min}[n]$, is the *envelope* of the challenge. The area of the envelope can be regarded as the variation of the R-impact of a certain challenge on a graph or network. More intuitively, that area quantifies the uncertainty or the amount of risk due to a challenge.

We use the metric framework as a methodology for our experiments: We monotonically increase the challenge to the network and measure the main metric parameters delay and delivery ratio. From the distribution we derive the metric envelope. As we will see, we use percentiles of the distribution rather than best and worst case.

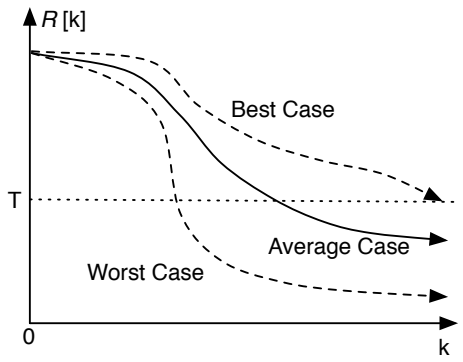



Figure 1: Metric envelope [Source: ResumeNet Deliverable D1.2]

# 4. EXPERIMENT

The experiments are performed in the ONE simulator [6] and using our own Space-Time-Graph methodology [7]. The ONE simulator was used to drive a discrete-event simulation. We chose ONE because it has been written explicitly for the simulation of opportunistic protocols and is used by many in the DTN community, rather than any other optimality reason. ONE provides implementations for various mobility models and routing protocols that we made use of.

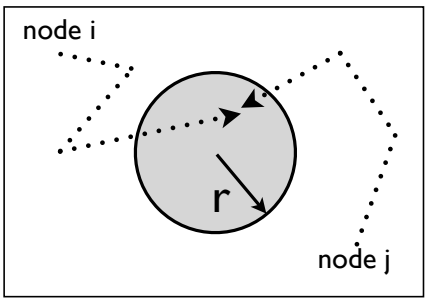
## 4.1 Challenge implementation

The three considered challenges, jammer, contact loss, and selfish node behavior are implemented in different ways. Common for all three implementations is that we used ONE to create mobility traces based on the given mobility models. These mobility traces are modified to reflect the challenge and feed either back into the ONE simulator through its external event interface, or into our Space-Time-Graph methodology.

The results are then obtained by analyzing the message delay and message delivery ratio in the result reports.

**Contact Loss** We randomly removed node contacts from the mobility trace and feed the modified trace back into the ONE simulator. The three routing protocol implementations EpidemicRouting, SprayAndWaitRouting and ProphetRouting were used.

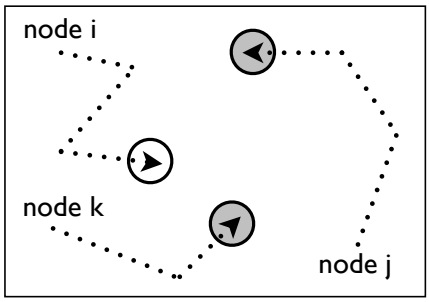
**Jammer** By extended ONE's logging functionality, we assign every event in the mobility trace the position of the involved nodes. Node contacts within range of the jammer are removed from the mobility trace and we feed the modified trace into the Space-Time-Graph methodology. The choice of the Space-Time-Graph methodology was to extend this methodology.




(a) Jammer

| contact id | involved nodes | | contact start time | contact end time |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| $C_0$ | 1 | 3 | 589 | 940 |
| ~~$C_1$~~ | ~~2~~ | ~~3~~ | ~~589~~ | ~~619~~ |
| $C_2$ | 6 | 7 | 639 | 699 |
| $C_3$ | 1 | 5 | 700 | 816 |
| ~~$C_4$~~ | ~~5~~ | ~~6~~ | ~~782~~ | ~~818~~ |
| ~~$C_5$~~ | ~~2~~ | ~~7~~ | ~~816~~ | ~~1185~~ |
| $C_6$ | 4 | 8 | 819 | 909 |
| $C_7$ | 8 | 6 | 938 | 1182 |
| ... | ... | ... | ... | ... |

(b) Contact loss




(c) Selfish nodes

Figure 2: Challenges

**Selfish Nodes** We extending the routing protocols in the ONE simulator the behave selfish, that is, probabilistically not copy or forward messages during node contacts (in the presented experiments, selfish nodes got assigned probability 1 to not copy and forward). Because nodes can only be assigned different behavior in ranges of their node identity, node selfishness based on node or betweenness centrality is implemented by permuting the node identity in the mobility trace which is fed into the ONE simulator. The two routing protocol implementations EpidemicRouting and SprayAndWaitRouting were modified and used.

The implemented challenges are illustrated in Figure 2.

## 4.2 Connectivity Traces

Our evaluation uses three connectivity traces of pairwise node with different characteristics. Two of the connectivity traces have been obtained from mobility models provided by the ONE simulator, the third reports Bluetooth sightings by groups of users carrying iMotes.

**Shortest Path Map based Mobility (SPMBM)** Nodes move between two randomly chosen locations by following the shortest path along connected segments representing roads on a map. In our case, a street map of Helsinki is used. The simulation area is 4500×3400 meters and includes 40 nodes.

**Random Waypoint (RWP)** Nodes move with random speed and random direction within a specified area, generating a trace with exponentially distributed contact times [8]. This trace does not have any social or spatial correlation between node contacts. We use the same size of the simulation area and number of nodes as with the Shortest Path Map based Mobility.

**Intel Trace** The node contacts are measured using Bluetooth motes [9]. The trace is collected for 9 nodes over the duration of 6 days and includes 2766 contacts.

## 4.3 Metric

Two metrics, delay and delivery ratio, are considered. For every delivered message we measure the delay to get it delivered to the destination node. The average, as well as the 25% and 75% percentile of the delay distribution are calculated and used for the metric envelope. For the metric envelopes we normalize the delay values with the average delay when no challenge is introduced.

We will see that it is important to consider delay and delivery ratio together because delay distributions for experiments with low delivery ratio are skewed towards small delay values when challenges are applied.

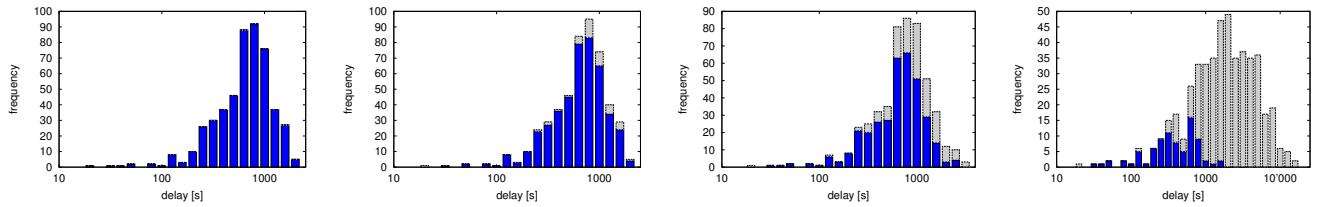
# 5. RESULTS

This section highlights a subset of the experimentation results best illustrating the resilience of opportunistic networks for different challenges, mobility traces, and routing protocols. We start the discussion with having a look at the delay distribution to understand the effect of challenges on individual messages and to point out the usefulness of metric envelopes compared to aggregated mean values.

## 5.1 Delay Distribution

The delay distribution gives more insight into how the randomly inserted messages spread in the network than a single aggregation value like the average delay. By using the same random seed in the simulation, we can study the impact of challenges and routing protocols by comparing the delay of individual messages in the different experiments. In the histograms in Figure 3 we color marked messages with an increased delay compared to unchallenged Epidemic routing grey, while messages whose delay did not get affected by the challenge are shown in blue.

Figure 3(b) and 3(c) both show the delay distribution for 20 selfish nodes for the Shortest Path Map based Mobility, with the difference that the 20 nodes are randomly selected in Figure 3(b) and according to node centrality in Figure 3(c). The selection of the selfish nodes has a clear impact on the delay. While the message delay is not significantly af-




(a) (b) (c) (d)

Figure 3: Histogram over the message delay. Messages that got delayed due to a challenge or different routing protocol are marked with grey color. (a) Epidemic routing; no challenge (b) Epidemic routing; 20 random selfish nodes (c) Epidemic routing; 20 selfish nodes according to node centrality (d) Spray and Wait routing; no challenge

fected by random selected nodes, the selection of the most central nodes[1] affects about 30% of the messages.

The selection of the routing strategy affects the delay of most messages, as seen in Figure 3(d). Messages that have a small delay with epidemic forwarding get affected least. This is natural as they get passed on during the first arising contacts that also Spray and Wait routing considers.

## 5.2 Jammer

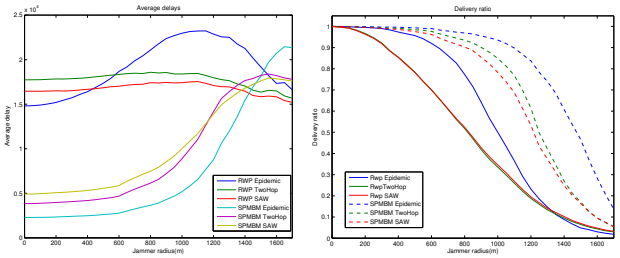
The jammer affects node contacts within its range such that no messages can be exchanged during these contacts. The results shown in Figure 4(a) show the average delay of delivered messages. For the shortest path map based mobility we observe good resilience against the jammer up to a range of about 800m where the delay starts to increase significantly up to a factor of 5 for epidemic routing. The different routing protocols show similar behavior.

For random waypoint mobility we see a difference in behavior between epidemic that increases with increasing jammer range, and two-hop and Spray and Wait routing that seems to be resilient to the jammer. Moreover, the average delay seems to decrease for a jammer range lager than 1000m. This observation is of course not due to a resilient behavior of the protocols – it is a statistical artefact: the average delay is calculated only over messages that actually are received within simulation time. Figure 4(b) shows the delivery ratio rapidly decreasing with increasing jammer range.

This is a fundamental issue of the chosen experimentation methodology. One could chose to only consider messages that are delivered for all jammer ranges, but that would in its turn only consider messages that have a short delay in the unchallenged case and result in an average delay significantly smaller than it actually is for the average message in the unchallenged case. The reason we present these results here is to point out the issue, it is neglected in the majority of simulation results and needs to be addressed by the research community.

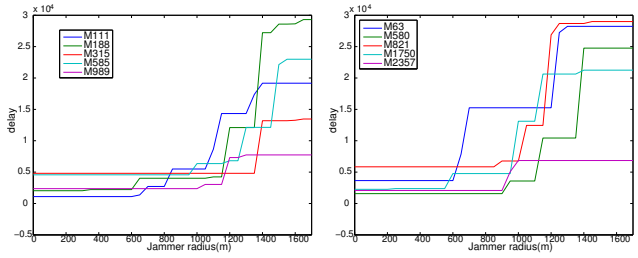
For this work, we suggest to only consider the delay for significant delivery ratios and thus a jammer range smaller than 500 meters. Furthermore, we illustrate in Figure 5 the effect of the jammer for five randomly selected messages that get received even for large jammer range.

[1]We observe similar results for both node centrality and betweenness centrality




(a) Average delay (b) Delivery ratio

Figure 4: Jammer — Average delay and delivery ratio for random waypoint (RWP) and Shortest Path Map Based Mobility (SPMBM)



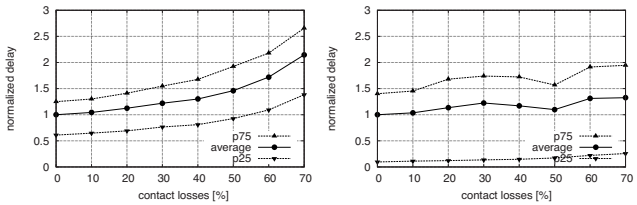

(a) Epidemic (b) Two Hop

Figure 5: Jammer — Delay of five randomly selected messages in Shortest Path Map based Mobility.

## 5.3 Contact Loss

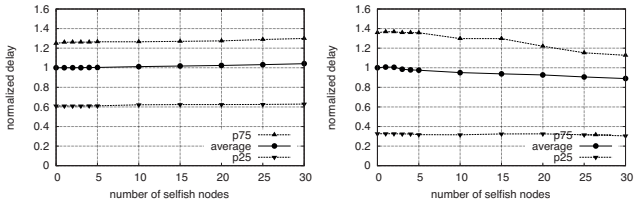
We now consider random contact losses not correlated to a location as in the case of a jammer. Figure 6 shows the normalized delay with the metric envelope defined by the 25% and 75% percentile. The average delay used for normalization is 850 seconds for the shortest path map based mobility, and 55'000 seconds for the Intel trace.

The two traces show different behavior. The Intel trace




(a) SPMBM (b) Intel trace

Figure 6: Contact loss for epidemic routing.




(a) Epidemic (b) Spray and Wait

Figure 7: Randomly selected selfish nodes.

has a significant space-time diversity with multi-node contacts, providing good resilience against contact losses. The average delay (and the distribution as such) does not significantly increase. Also, the 25% percentile is low, suggesting that the average delay is dominated by a few messages with high delay. The 25% percentile increases by a factor of 2 for 50% contact loss, and a factor of 3 for 70% contact loss. The shortest path map based mobility, in contrary, shows an exponential increase of the average delay similar to the case of the jammer.

## 5.4 Selfish Nodes

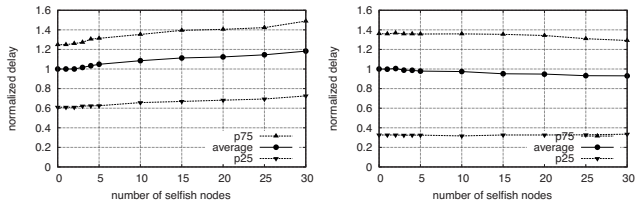
The experiments for selfish node behavior are performed with shortest path map based mobility and we compare the impact of the selection of selfish nodes: random, according to node centrality, and according to betweenness centrality.

*Random Centrality.*

Figure 7 shows that epidemic routing is resilient against selfish nodes at least up to 30 selfish nodes out of the 40 nodes. This result is in line with the analytical results presented in [10]. Spray and Wait routing even shows an improvement in the delay, while maintaining the delivery ratio, suggesting that the early contacts used without challenge are not necessarily on the fastest space-time path. The average delay used for normalization is 850 seconds and 2900 seconds for epidemic routing and Spray and Wait routing, respectively.

*Node Centrality.*

Strategically assigning nodes with most contacts to be selfish makes the network less resilient against the challenge.




(a) Epidemic (b) Spray and Wait

Figure 8: Selfish nodes selected on node centrality.

Figure 8 shows that the average delay increases 20% for 30 selfish nodes out of 40 nodes for epidemic routing. Again, for Spray and Wait the delay is slightly decreasing with an increasing number of selfish nodes.

*Betweenness Centrality.*

The results for betweenness centrality are almost identical to node centrality, although the nodes with highest betweenness centrality are not the same nodes that have highest node centrally in our trace.

# 6. CONCLUSIONS

We find that the diversity in space-time paths give a high degree of resilience to contact and node failures, offering alternative paths to deliver data. Although impact of a challenge on an individual message can be significant if a contact opportunity is missed, we find that the overall performance decrease is typically graceful until a significant amount of the node contacts or nodes are affected, depending on the diversity offered by the topology.

The behavior is similar for most of the studied forwarding protocols, with the exception of the Spray and Wait routing protocol which surprisingly even showed cases where the data delivery delay improved under challenges affecting node contacts.

We show on the example of node selfishness that selective challenges affecting strategic nodes and contacts have a higher impact on the system performance but, depending on the mobility trace, increases the delay with less than 20% compared to random selection.

# 7. REFERENCES

[1] M. Scholler, P. Smith, C. Rohner, M. Karaliopoulos, A. Jabbar, J. P. Sterbenz, and D. Hutchison, "On realising a strategy for resilience in opportunistic networks," in *Proceedings of the Future Network and MobileSummit 2010 Conference*, 2010.

[2] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE MCCR*, 2003.

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based

Disruption-Tolerant Networks," in *IEEE Infocom*, 2006.

[4] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay tolerant networks," in *in Proc. ACM MobiHoc*, 2008.

[5] T. Spyropoulos, K. Psounis, and C. 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*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[6] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 55:1–55:10. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674

[7] M. Karaliopoulos and C. Rohner, "Trace-based performance analysis of opportunistic forwarding under imperfect cooperation conditions," in *Proceedings of the INFOCOM 2012 mini-conference*, 2012.

[8] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[9] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD trace cambridge/haggle/imote/intel (v. 2006-01-31)," Jan. 2006.

[10] M. Karaliopoulos, "Assessing the vulnerability of dtn data relaying schemes to node selfishness," *Communications Letters, IEEE*, vol. 13, no. 12, pp. 923 –925, december 2009.

# Paper D

# Congestion Avoidance in a Data-Centric Opportunistic Network

**Fredrik Bjurefors, Per Gunningberg, and Christian Rohner**

ACM SIGCOMM Workshop on Information Centric Networking ICN11, Toronto, 2011

Abstract: In order to achieve data delivery in an opportunistic network, data is replicated when it is transmitted to nodes within communication reach and that are likely to be able to forward it closer to the destination. This replication and the unpredictable contact times due to mobility necessitate buffer management strategies to avoid buffer overflow on nodes. In this paper, we investigate buffer management strategies based on local forwarding statistics and relevance of the data for other nodes. The results obtained on our emulation platform for opportunistic networks show that strategies with a high data refresh rate achieve the most efficient delivery and generate the smallest overhead on our community and mobility scenarios.

# Congestion Avoidance in a Data-Centric Opportunistic Network

Fredrik Bjurefors
Uppsala University, Sweden
fredrik.bjurefors@it.uu.se

Per Gunningberg
Uppsala University, Sweden
per.gunningberg@it.uu.se

Christian Rohner
Uppsala University, Sweden
christian.rohner@it.uu.se

Sam Tavakoli
Uppsala University, Sweden
sam@tavakoli.se

## ABSTRACT

In order to achieve data delivery in an opportunistic network, data is replicated when it is transmitted to nodes within communication reach and that are likely to be able to forward it closer to the destination. This replication and the unpredictable contact times due to mobility necessitate buffer management strategies to avoid buffer overflow on nodes. In this paper, we investigate buffer management strategies based on local forwarding statistics and relevance of the data for other nodes. The results obtained on our emulation platform for opportunistic networks show that strategies with a high data refresh rate achieve the most efficient delivery and generate the smallest overhead on our community and mobility scenarios.

## Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: General—*Data communication*

## General Terms

Experimentation, Measurement

## Keywords

Haggle, Data-centric, Opportunistic Networks, Delay Tolerant Networks, Congestion Avoidance

## 1. INTRODUCTION

Communication in opportunistic networks is based on sporadic and intermittent contacts between mobile nodes. These contacts are used to exchange data with nodes that are likely to move data closer to the destinations. The lack of end-to-end paths makes data-centric networking with a focus on content rather than on location an attractive communication model for opportunistic networks.


*ICN'11,* August 19, 2011, Toronto, Ontario, Canada.


Considerable work has been done on forwarding strategies, deciding what data to replicate and exchange during contacts. Knowledge about movement pattern, social relations in communities and user interests can be used for efficient data dissemination within and between communities. Identification of central nodes with many contacts, called hubs, and nodes that have frequent contacts with two or more communities, called bridge nodes, have been shown to be efficient in data dissemination [5].

Data replication in general leads to "node congestion," i.e., filling up buffers with replicated data. Hubs and bridges in particular are likely to face the issue of evicting data to leave space for new data. In this paper, we emphasize the importance of efficient congestion avoidance. We investigate into buffer management and data dropping strategies based entirely on local information, and show that strategies with a high data refresh rate achieve the most efficient delivery and generate the smallest overhead in our community and mobility scenarios.

The paper is structured as follows. In Section 2, the data-centric architecture used in this work is described. In Section 3, congestion avoidance strategies in terms of buffer management are discussed. Section 4 describes the methodology and the scenarios. Section 5 describes and discusses the results of our experiments with different buffer management strategies. Related work is reviewed in Section 6. Last, we conclude our findings in Section 7.

## 2. HAGGLE

We evaluate congestion avoidance using the Haggle [10, 11] test-bed. Haggle is a data-centric network architecture, designed for mobile nodes, based on a publish/subscribe model. A Haggle node can opportunistically take advantage of any communication technology of mobile devices, such as Wi-Fi and Bluetooth. A device runs an instance of Haggle to which applications connect. The instance becomes a node in an opportunistic Haggle network.

### 2.1 Data Objects and Node Descriptions

A Haggle node has a buffer management system where it stores *data objects*. Data objects consist of *data* with associated *metadata*. Data could be anything from a couple of bytes to a large video clip of many Mbytes. Metadata uses *attributes* that describes the actual data. Nodes also express their *interests* in data in the form of these attributes. A *node description* consists of metadata describing what the node is

interested in and contains other node-specific information. An application on a node generates new data objects as well as consumes data objects that it is interested in.

## 2.2 Forwarding

When two nodes are getting in contact, they first exchange their node descriptions. The interest attributes of the two nodes are matched against metadata attributes of the data objects stored on the nodes. If there is a match of interests, data is exchanged. In this manner, data objects are eventually spread among all nodes with common interests.

In addition, nodes exchange data that the other node is not interested in by itself, thus acting as a relay node. The nodes calculate which other nodes in the network the the neighboring node is more likely to meet in the future. If the other node is more likely to be able to forward the data towards the destination than the node itself, the node pushes the data objects to the neighboring node.

The relayed data objects will occupy space in the *relay buffer*. The objects will stay there and will be replicated and exchanged when meeting other nodes with interest in these data objects. Data objects are replicated when they are forwarded, i.e., forwarding does not mean that a data object is deleted from the sending node's data store. Eventually all buffers will be filled up and there must be a strategy to evict data objects to create space for new data objects.

# 3. CONGESTION

In an opportunistic network, data dissemination is done through data replication. Multiple copies of data increase the chance for data to be delivered, but it also congests the network. The remedy in place to avoid congestion is to limit the number of copies of data in the network.

A node receiving many data objects during a short period of time will drain its storage resources. When storage is depleted, a node will not be able to receive or relay objects, which means that the whole network forwarding efficiency will be hampered by these blocked nodes. To get the delivery system to work properly under short overload periods congestion avoidance methods are needed, and when the buffers are filled up, there is a need for an algorithm for dropping objects.

Traditional congestion avoidance systems, like TCP use feedback information from the end destination to the source (e.g., ACKs) and the sender transmission rate is throttled to the capacity of the end-to-end path. TCP is complemented with an algorithm to selectively drop segments in routers, e.g., RED [4]. Opportunistic networks lack an end-to-end path and the whole data object is forwarded to the destination on a store-carry-forward basis without an ACK back to the sender. Therefore, traditional congestion avoidance techniques cannot be used.

The objective with congestion avoidance in this type of opportunistic system is to take decisions on how much data to replicate, and what to drop from the relay buffer and when, with minimum impact on the overall delivery ratio. Since there is no feedback information, the decision to drop must be based on local information that a node can gather when it is in contact with other nodes.

The information that is available to a node is forwarding statistics and node descriptions. Forwarding statistics consist of the number of times a node has replicated a data object. Also, the forwarding probability of the node that the data object is sent to is collected. A highly replicated object is a candidate for dropping since there are already many of them in the network. The node descriptions received from other nodes give information about the subscribers' interests. An object in which there is little interest is also a candidate for dropping.

## 3.1 Buffer Management

Haggle selects the data objects to be dropped from the relay buffer based on age. Data objects older than a certain threshold will be dropped. Aging is performed periodically. The chosen age threshold and periodicity, in relation to the rate of incoming data defines the occupation in the relay buffer and may fluctuate significantly over time.

In this work, we take buffer size in consideration and introduce a limit on the relay buffer. If the relay buffer on a node gets full, data objects will be evicted from the buffer. Resource management in a node considers all data objects that are "not of interest" and removes them. The size of object is not taken into account in this first evaluation but it could also be used as a factor. A large object releases more buffer space but a large object takes more time to spread due to limited bandwidth and contact times.

### *3.1.1 Evaluated dropping strategies*

We evaluate data object dropping strategies based on interests and on the degree of replication.

**LI – Least Interested**: Drop the data object that the least number of neighbors are interested in. This strategy has two side effects: (a) it reduces the diversity of content since the object will eventually be extinct if no one is interested in it for a long time, (b) but it increases the overall delivery ratio of other objects since the overall interest matching is increasing.

**MI – Most Interested**: Drop the data objects that most neighbors are interested in. As opposed to the previous strategy, this will maintain data object diversity in the network, but may have a negative effect on the delivery ratio. The strategy will reduce the number of copies of data objects that are most looked for. On the other hand, a large number of copies of the same data were shown not to increase the delivery ratio [3].

The following strategies drop data objects based on local forwarding history, i.e., replication of data.

**Max – Max Copies**: Drop data objects after a max number of copies have been made at the local node. Data objects are removed after they have been copied *Max* number of times. This makes the relaying node reject relayed data objects until space has been cleared in the buffer, no preemption is done.

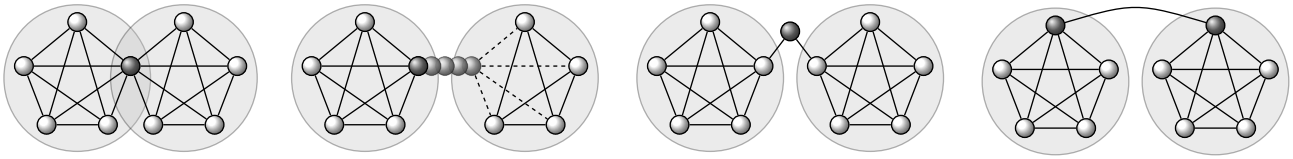
**MF – Most Forwarded**: Drop the data object with the highest number of replications made at the local node.

**LF – Least Forwarded**: Drop the data object with the lowest number of replications made at the local node.

**Random**: Drop randomly chosen data objects from the relay buffer. We compare the other five strategies against this reference case.

**Infinite Buffer**: Reference best case. It is the overall best case, represented by nodes with an infinite relay buffer. No object needs to be dropped

**No Buffer**: Reference worst case. The worst case is represented by a node strategy, which only accepts data that the node itself is interested in, i.e., there is no relay buffer.



(a) HN – Hub Node (b) DM – Data Mule (c) BN – Bridge Node (d) CH – Cluster Head

**Figure 1: In each scenario five communities are connected in a chain. Depicted are the four different hub and bridge nodes (shown in dark gray) used in the scenarios. Lines between nodes represent intermittent connects.**

## 3.2 Flow Control

Flow control prevents a sender from overwhelming a receiver with data by having the receiver signal its ability to handle incoming data. Haggle uses two combined mechanisms to control the flow of objects between nodes. The first is to limit the number of data objects exchanged. A receiving node must have at least that buffer space. The second is to limit the exchange of data objects to only the highest ranked with respect to interest. In general, a fixed limit will decrease the overall dissemination of data objects. It will have the same effect on dissemination as a network with fewer contact opportunities. If the network is sparsely connected, only the highest ranked data objects are likely to be exchanged.

The maximum number of data objects possible to receive and a ranking threshold can be advertised in the node description. However, node descriptions are only exchanged during the initial phase of a contact and there is no dynamic feedback during the actual transfer that can throttle the sender.

By limiting the number of shared data objects, we can only reduce the risk of congestion. It is not possible to avoid or recover from congestion. For that purpose, dropping strategies are needed.

# 4. EVALUATION

The performance evaluation is performed on a test-bed based on virtual computers running Linux. One virtual machine hosts one Haggle node. The virtual computers are running on top of Xen [2], a virtualization software that connects the nodes through an Ethernet bridge where we can filter traffic between the nodes. The mobility, community graphs and contact times are emulated and are filtered according to scenarios and connectivity traces. The user data production and consumption for each node is identical for each scenario in order to be able to compare strategies.

## 4.1 Scenarios

Four scenarios are used in the evaluation. In each scenario, five communities are connected in a chain, unless mentioned otherwise. A community is a group of nodes that have many contact opportunities between each other [5]. In the scenarios we vary the way neighboring communities are connected to each other, as depicted in Figure 1, and we study the impact of different congestion strategies.

For all scenarios, we use a community size of five nodes where each node have intermittent contact with all other nodes in the community. Contacts occur according to a Markov model with a mean contact time of 60 seconds and an inter-contact time of 240 seconds. The total length of an experiment is one hour.

The following hub or bridge node(s) scenarios are used:

**HN − Hub Node**: Two communities are connected by a hub node. All data objects exchanged between two communities have to go through this hub.

**DM − Data Mule**: A "data mule" node belongs to two communities but is only present in one community at a time. A data mule receives all packets destined to a neighboring community. During a stay in one community, the mule will receive packets destined to nodes in a neighboring community. The buffer may become full before it leaves for the other community, therefore, data must be dropped. A "data mule" node stays in a community for 600 seconds before it moves over to the other community.

**BN − Bridge Node**: In this scenario communities are connected by a bridge node between each community pair. Three nodes are likely to be congested, the bridge node and the two nodes connecting to the bridge. In this scenario we interconnect four communities instead of five to keep the number of nodes comparable with the other scenarios.

**CH − Cluster Head**: One node from each community is connected to a node in the neighboring community, forming a link between them. Congestion occurs by the fact that a node within a community that wants to send data to a node in any other community must go through the cluster head node of the community, thus a bottleneck occurs.

## 4.2 Scenario parameters

Each node is set to be interested in five attributes. They are chosen from a pool of $N = 100$ attributes. The attributes are assigned to the nodes according to a Zipf law distribution of $\alpha = 0.368$ [8]. Nodes are set to have 200 data objects stored in their buffer, marked as received from the node's application. Each data object is also assigned five attributes from the same pool of 100 attributes. These attributes are chosen from a uniform distribution.

In our experiments, we vary the size of the relay buffer. During each contact the nodes exchange up to 20 data objects matching their own interest and up to 20 data objects that are relayed. When the relay buffer reaches 80% utilization, data objects will be removed according to the dropping strategy. We limited the buffer to 80% in order to let the chain of events in the Haggle kernel to take place without risking a buffer overflow. When a relayed data object also is of interest for the relaying node the data objects will not take up space in the relay buffer. Instead, it will be put in another buffer that is considered infinite. Aging is not used in our experiments.

| | Inf.Buffer | | | | | MF buffer=25 | | | | | MF buffer=100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Community No. | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| Hub Node | 93 | 51 | 39 | 26 | 22 | 100 | 46 | 34 | 20 | 17 | 86 | 50 | 41 | 31 | 31 |
| Data Mule | 90 | 46 | 17 | 7 | 2 | 95 | 40 | 12 | 4 | 0 | 94 | 43 | 16 | 5 | 2 |
| Bridge Node | 100 | 23 | 6 | 2 | | 100 | 10 | 2 | 1 | | 100 | 13 | 4 | 0 | |
| Cluster Head | 100 | 36 | 18 | 7 | 8 | 100 | 29 | 18 | 12 | 9 | 100 | 27 | 16 | 12 | 12 |

**Table 1: Delivery ratio per source community for nodes in community one. Community number five represents the community furthest away from community one.**

# 5. RESULTS AND DISCUSSION

In this section, we present performance results from our experiments. We report on the metrics such as delivery ratio, dissemination speed, and forwarding overhead.

## 5.1 Delivery ratio

We measure the delivery ratio per node. With this, we mean how many of the nodes in the network that have an interest in the object, will actually get a copy within a time frame. The *delivery ratio* for the whole network is thus the delivery ratio over all nodes and all objects. Figure 2 shows the delivery ratio in all four scenarios and the dropping strategies. As can be obsverd, the Hub Node scenario is performing the best for all dropping strategies. This shows the importance of good and frequent connectivity of the hub node within and between communities. The other scenarios perform significantly worse with respect to delivery ratio with the Cluster Head scenario slightly better than Data Mule and Bridge Node scenarios as it provides a more direct connection between the communities.

The relative performance between the strategies are similar over the different scenarios. The strategies Most Forwarded (MF) and Random are performing closest to the reference strategy Infinite Buffer. The MF strategy (which drops the data object with the highest number of replications made at the local node) performs the best. An explanation for this is that it favors diversity so that the not so popular objects get a chance to be spread to the interested nodes. The Least Forwarded (LF) strategy that drops the data object with the lowest number of replications has the opposite effect by favoring popular objects so that they are quickly spread in the network. On the other hand, it creates many copies of the same objects, which takes space in buffers and is not efficient from the whole network-spreading point of view. Hence, the delivery ratio, which is calculated over all objects and all nodes is hampered. The performance of MI, to drop the objects with most interest, follow the same discussion as for the MF—the objects with the most interest in are likely to be the most copied as well. That Random dropping is doing so well can again be argued with the same rationale—it allows less popular objects to get space in the buffers. The dropping strategy Max Copies is performing close the worst case strategy No Buffer. This is an indication that it pays to really use a dropping strategy that often exchange data in the buffer.

### 5.1.1 Hub Node scenario

The Hub Node scenario has the highest delivery ratio because of the well-connected hub node that resides within two communities. We will take a closer look at Hub Node with respect to its impact on buffer size and dissemination speed, as well as forwarding overhead. The other scenarios show similar results, but with a lower delivery ratio.

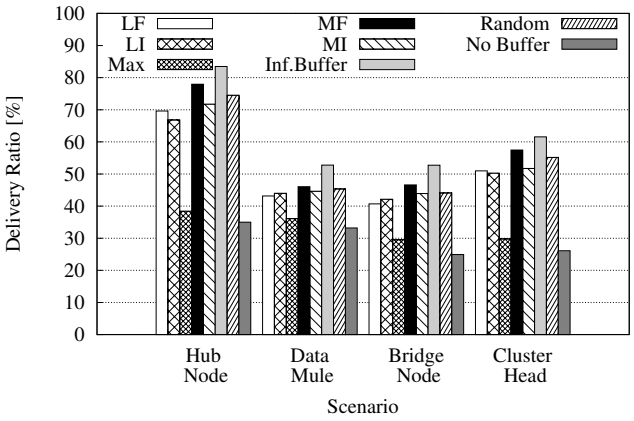



**Figure 2: Delivery ratio using six object-dropping strategies in four scenarios. The dropping strategies are: Least Forwarded (LF), Least Interested (LI), Max Copies made of data, Most Forwarded (MF), Most Interested (MI), and Random. The cases, Inf. Buffer do not drop any packets and No Buffer which represents the case when nodes do not relay objects.**

Figure 3 shows the delivery ratio for the different dropping strategies with varying buffer sizes. We observe a clear difference between the Max Copies and the other strategies. While the other strategies achieve a fair delivery ratio already with small buffer size, the Max Copies strategy scales linearly with the buffer size between the two reference strategies No Buffer and Infinite Buffer. For the other strategies, it is observed that they scale with buffer size up to 100 data objects from which point a larger buffer size does not improve the delivery ratio until it approaches the maximum needed buffer occupation.

For this particular contact scenario, our interpretation is that a buffer of 100 data objects is enough if there is a dropping strategy. In other words, above this buffer size the dropping strategy is more significant than an increase in buffer size. Furthermore, the Max Copies strategy is not suited for scenarios where nodes only have a few possible neighbors that may share the interests. What happens in this strategy is that objects in the relay buffer will not be discarded until they have been replicated and sent to other nodes. The consequence is that objects that few nodes are interested in will occupy buffer space, blocking new objects that may have a better chance to be spread.

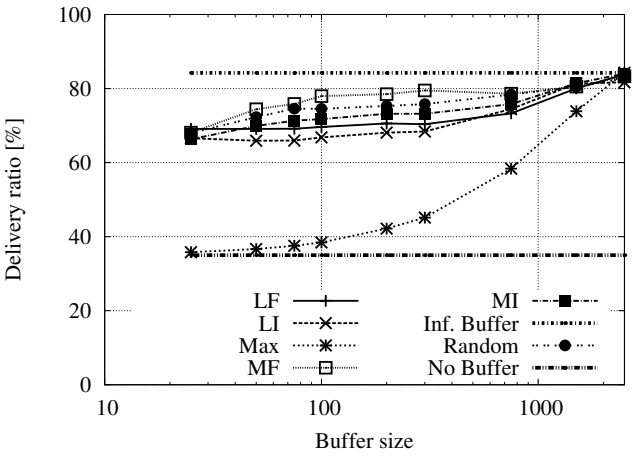



**Figure 3: Delivery ratio using different dropping strategies, in the Hub Node scenario.**

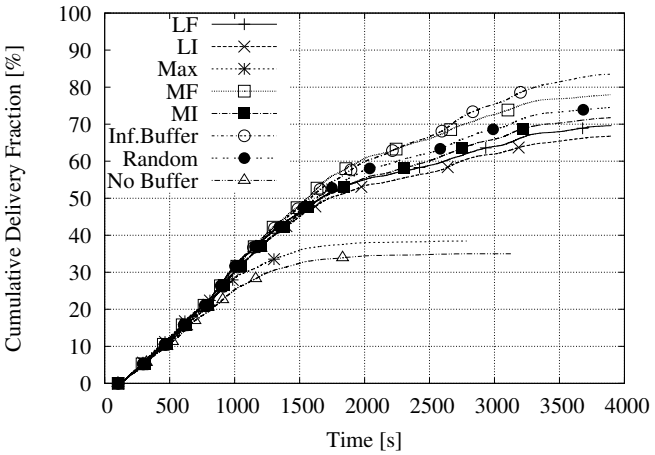



**Figure 4: Cumulative delivery fraction over time in the Hub Node scenario using a buffer size of 100 data objects.**

## 5.2 Dissemination speed

Table 1 shows the delivery ratio from community 1-5 respectively, to nodes in community 1. It shows how many data objects that nodes in community 1 will get from the other communities given the nodes' interests in the objects. Since objects are forwarded from community to community via relay nodes, the delivery rates are implicit indicators of the dissemination speed. As can be seen in the table, the nodes of community 1 in the Hub Node scenario receive 93% of all objects of interest in community 1 and 22% of the objects of interest in community 5. The Infinite Buffer strategy provides the fastest dissemination as discussed earlier, but even for a small buffer size (25), the dissemination speed is not too far from the best case. When comparing the performance of interconnection alternatives, the Hub Node is doing better than the others in terms of the delivery ratio, i.e., it has the fastest dissemination speed.

Figure 4 shows the cumulative delivery ratio for all dropping strategies with respect to time. The MF strategy comes very close to the Infinite Buffer reference case for all times. Thus, it is the best dropping strategy when optimizing for dissemination speed or shortest average delay.

If a No Buffer strategy is used, data objects will only be exchanged when nodes are in contact and share interests. There is no relaying of objects. As can be seen in the figure, objects will eventually stop being exchanged when buffers for shared interests are exhausted. Even with a small relay buffer and a dropping strategy, data objects will get through to the interested nodes, but at the cost of forwarding overhead.

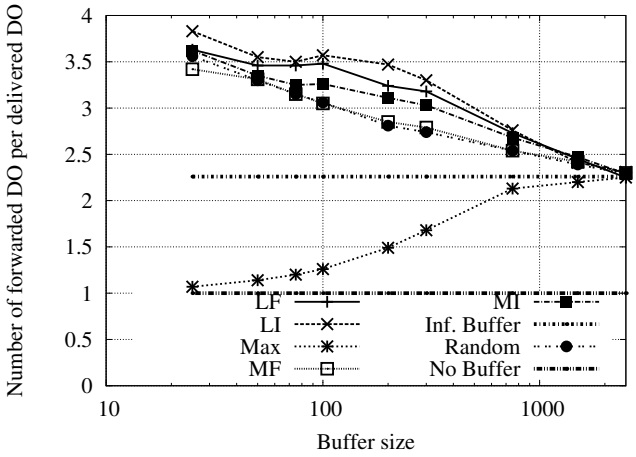



**Figure 5: Overhead in terms of average number of forwarded data objects per successfully delivered data object, in the Hub Node scenario.**

## 5.3 Overhead

With overhead, in this paper, we mean the number of times a data object has to be forwarded per delivery to an interested node. The system creates copies of objects that are distributed to the interested nodes. The more copies of an object that are circulating the more likely it is that at least one of them will reach a node interested in it. Likewise, the average delay to a node will be shortened with more copies if there is enough capacity in the system to accommodate all copies. Therefore, it is interesting to understand how many times forwarding takes place per efficient delivery of an object, and which dropping strategy is the most efficient.

In contrast to flooding and epidemic spreading where nodes exchange copies every time they meet, an interest-based system only exchanges copies when there are matching interests or when there is a relaying opportunity. The nodes will store the copies until the right time and not resend them as soon as possible, which is the case in flooding and epidemic spreading. This conserves both bandwidth and energy but requires larger buffers.

Figure 5 depicts this overhead per dropping strategy and buffer size. The most efficient strategy and the reference case is that of Infinite Buffers. In this strategy, a node always has buffer space to receive an object and can store everything it receives. There is no resending of objects due to dropping of objects. Here we can see that a data object is copied on average 2.2 times to reach nodes that are interested in

it. When studying the graphs for dropping strategies, we observe that they use between 3.8 and 2.2 copies depending on strategy and buffer size. To achieve a high delivery ratio, a data object has to be resent after it has been dropped from the buffer. The difference to the Infinite Buffers case of 2.2 is due to the dropping strategy, when the dropped objects have to be copied again. With a smaller buffer size, more objects have to be dropped and replaced than for bigger buffer sizes. The No Buffer strategy does not have any relay buffer. The object is copied directly to the interested party. MF has the lowest overhead compared to other dropping strategies with a high delivery ratio. MF is also the strategy with the highest delivery ratio. To summarize, storage space is traded against forwarding overhead and with less objects dropped, less objects have to be resent.

# 6. RELATED WORK

Related congestion management work has been done in the area of host-centric Delay Tolerant Networking. In DTN, a node guarantees to forward a bundle (message) to a new custodian or deliver it to the destination. When a bundle is accepted, it cannot be dropped since there is no copy left at the sender. Seligman et al. propose to handle storage congestion at custody nodes by migrating bundles to alternative custodians [12]. Instead of migrating already stored bundles, Zhang et al. [13] propose a congestion management scheme that takes active decisions on whether to accept a custody bundle to avoid congestion. In our Haggle system, there is no commitment of guaranteed delivery, data objects are replicated in each forwarding step.

Balasubramanian et al. [1] and Krifa et al. [7, 6] have studied routing in a DTN as an allocation problem to avoid congestion and possible dropping of a bundle. Joint scheduling is done by using global information. Global information is either propagated through the network [1] or acquired by statistical learning [7, 6]. Both proposals can optimize dissemination to a specific metric, e.g., average delay or delivery probability.

Lindgren et al. [9] evaluate queuing policies based on local forwarding counters and transient forwarding probabilities, using PRoPHET. They show that probabilities can be used in buffer management to increase the delivery ratio. With weights reflecting the number of copies they also show how PRoPHET can be made more energy efficient per packet delivery. An evaluation is performed in a host-centric and intermittently connected network.

# 7. CONCLUSIONS

We have evaluated dropping strategies at congested nodes for data-centric opportunistic networks in different community scenarios. In such networks congestion avoidance, buffer handling, and choice of data object to drop must be based on local information. Nodes do not have complete information of the interests of other nodes in the network since Node Descriptions will only spread between nodes that share interests.

From the experimental evaluation, we conclude that the MF strategy (which drops the data object with the highest number of replications made at the local node) performs overall best with respect to delivery ratio, delay, and overhead at a congested node, i.e., when there is not enough buffer space for relaying objects. Furthermore, using local replication information when dropping gives a higher delivery rate compared to using local interest information. The strategy to randomly select a data object to drop comes surprisingly close to the best cases in our measurements. This shows that already with a small relay buffer, performance gains can be achieved with a dropping strategy where data objects often are exchanged.

# 8. ACKNOWLEDGMENTS

This research was funded by the ResumeNet project under the EU grant FP7-224619 and the Haggle project under the EU grant IST-4-027918.

# 9. REFERENCES

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, 2007.

[2] P. R. Barham, B. Dragovic, K. A. Fraser, S. M. Hand, T. L. Harris, A. C. Ho, E. Kotsovinos, A. V.S. Madhavapeddy, R. Neugebauer, I. A. Pratt, and A. K. Warfield. Xen 2002. Technical Report UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, January 2003.

[3] F. Bjurefors, P. Gunningberg, E. Nordström, and C. Rohner. Interest dissemination in a searchable data-centric opportunistic network. In *European Wireless Conference, 2010*, pages 889–895, 2010.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1:397–413, August 1993.

[5] P. Hui and J. Crowcroft. Bubble rap: Forwarding in small world DTNs in ever decreasing circles, 2007.

[6] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal Buffer Management Policies for Delay Tolerant Networks. In *SECON*, page 10, 2008.

[7] A. Krifa, C. Barakat, and T. Spyropoulos. An optimal joint scheduling and drop policy for delay tolerant networks. In *WoWMoM*, pages 1–6. IEEE Computer Society, 2008.

[8] V. Lenders, G. Karlsson, and M. May. Wireless ad hoc podcasting. In *SECON*, June 2007.

[9] A. Lindgren and K.S. Phanse. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *COMSWARE, 2006*, pages 1–10, 2006.

[10] E. Nordström, P. Gunningberg, and C. Rohner. Haggle: a data-centric network architecture for mobile devices. In *MobiHoc S3 '09*, pages 37–40. ACM, 2009.

[11] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A networking architecture designed around mobile users. In *IFIP WONS 2006*, 2006.

[12] M. Seligman, K. Fall, and P. Mundur. Alternative custodians for congestion control in delay tolerant networks. In *CHANTS '06*, pages 229–236. ACM, 2006.

[13] G. Zhang, J. Wang, and Y. Liu. Congestion management in delay tolerant networks. In *WICON '08*, pages 1–9. ICST, 2008.

# Paper E

# Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks

**Christian Rohner, Fredrik Bjurefors, Per Gunningberg, Liam McNamara, and Erik Nordström**

Submitted to ACM Workshop on Mobile Opportunistic Networking MobiOpp2012, Zürich, 2012

Abstract: Opportunistic networks use unpredictable and time-limited inter-node contacts to disseminate data. Therefore, it is important that protocols transfer useful data when contacts do occur. Specifically, in a data-centric network, nodes benefit from receiving data relevant to their interests. To this end, we study five strategies to select and order the data to be exchanged during a limited contact, and measure their ability to promptly and efficiently deliver highly relevant data. Our trace-driven experiments on an emulation testbed sug- gest that nodes benefit in the short-term from ordering data transfers to satisfy local interests. However, this can lead to suboptimal long-term system performance. Only sharing locally relevant data can also lead to networks being segregated due to divergent interests. A non-local understanding of other nodes' interests is necessary to effectively move data across the network. If ordering of transfers for data relevance is not explicitly considered performance is comparable to random, which limits the delivery of individually relevant data.

# Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks

Christian Rohner
Uppsala University, Sweden
christian.rohner@it.uu.se

Fredrik Bjurefors
Uppsala University, Sweden
fredrik.bjurefors@it.uu.se

Per Gunningberg
Uppsala University, Sweden
per.gunningberg@it.uu.se

Liam McNamara
Uppsala University, Sweden
liam.mcnamara@it.uu.se

Erik Nordström
Princeton University, USA
enordstr@cs.princeton.edu

## ABSTRACT

Opportunistic networks use unpredictable and time-limited inter-node contacts to disseminate data. Therefore, it is important that protocols transfer useful data when contacts do occur. Specifically, in a data-centric network, nodes benefit from receiving data *relevant* to their interests. To this end, we study five strategies to *select and order* the data to be exchanged during a limited contact, and measure their ability to promptly and efficiently deliver highly relevant data.

Our trace-driven experiments on an emulation testbed suggest that nodes benefit in the short-term from ordering data transfers to satisfy local interests. However, this can lead to suboptimal long-term system performance. Only sharing locally relevant data can also lead to networks being segregated due to divergent interests. A non-local understanding of other nodes' interests is necessary to effectively move data across the network. If ordering of transfers for data relevance is not explicitly considered performance is comparable to random, which limits the delivery of individually relevant data.

## 1. INTRODUCTION

In opportunistic networks, mobile nodes exchange data when they are within wireless communication range. Data items are disseminated via neighbours to enable them to reach interested parties, with nodes using the *store-carry-forward* principle to collaboratively establish a data relaying service.

Due to the unpredictability of node contact duration it is not known how many successfull item transfers will be performed during a contact. Therefore, a data dissemination protocol must judiciously decide *what* data to exchange and in *which order*, to make the best use of a time-limited contact and to achieve the best "value" for the network at minimal cost.

Transferring an item of data represents a selection to spread that particular item instead of another. The receiver then gains the utility from the transferred items and can also spread it to other nodes in the network. When choosing a dissemination system, a key consideration is balancing local benefit to the receiver and potential global benefit for the entire system.

For the rest of the paper, we shall use the following terminology to refer to two aspects of opportunitic data dissemination. Deciding whether an item should be transferred to a given host is a *selection* decision, deciding which order the selected items should be sent in is called an *ordering* decision.

Numerous selection algorithms have been proposed to efficiently choose nodes that will carry data to other nodes in the network [9, 11, 15, 18, 23]. These decisions are often made based on node properties, for example, contact history or role in community structures. Deciding which items to drop from a finite data item buffer has been extensively studied, with strategies based on FIFO, LIFO, TTL, or age [16, 22] of items. By design, however, these strategies see each delivered data item as contributing the same value to the network, and therefore strive to make a similar effort in delivering each item without worrying about ordering. In reality, however, an item's contributed value to different nodes will vary depending on some measure of *relevance*.

In this paper, we claim that all data items are not created equal, and that dissemination systems can benefit from reflecting this reality. One mechanism for a host to determine the relevance of a data item is based on the match between its interests and the item metadata. A node could then decide which items to select for transfer based on the perceived benefit to the recipient or the network as a whole. As the amount of selected data that can be transferred between two nodes is limited by the duration of their contact, the order of the data transferred plays an important role for the performance of the dissemination.

With these observations in mind, we study five data-centric ordering strategies and their ability to deliver relevant data to nodes. By matching user interests against content metadata, recipients assign each data item a relevance score. We then judge a dissemination strategy mainly by its ability to deliver the higher scored data before the lower scored.n an efficient manner.

We evaluate each dissemination strategy using the Haggle

network architecture [20] running on a trace-driven emulation testbed. Our main results are not specific to Haggle; we simply take advantage of the functionality offered by the platform. We use three different contact traces to study the behaviour of each strategy under varying environments, in terms of connectivity and community structure. Our results show that data transferring based on the relevance leads to a more efficient dissemination of data.

The rest of the paper is organised as follows: §2 considers related work, §3 presents our formulation of relevance-driven data dissemination. Our experimental details are specified in §4 followed by the results in §5. Finally, our conclusions are discussed in §6.

## 2. RELATED WORK

A number of data-centric dissemination systems have previously been proposed that use "content channels" as their underlying dissemination mechanism [7, 13, 14]. Data items are classified into channels that users subscribe to and then synchronise when in contact. Content channels provide a coarse definition of relevance, and there is no intentional ordering of data within channels or when synchronising.

Moghadam et al. [17] proposed an interest-aware content distribution protocol with a more fine grained definition of relevance, similar to our relevance score. However, they only selected whether to forward or not, and not how to order data.

Numerous forwarding algorithms have been proposed that use context or social knowledge to enhance the dissemination [4, 5, 9, 21]. However, the goal of such protocols is to compute good "data mules", and are otherwise agnostic to the relevance of the data forwarded.

Forwarding decisions and buffer management in opportunistic networks have been extensively studied. It has been shown that buffer management strategies, *e.g.*, FIFO, LIFO, "most forwarded" or "oldest" can have a large impact on the delay and overhead in a DTN [16]. However, these strategies optimise for high delivery ratio or low delay, and have at most an *incidental* effect on the relevance of the data delivered.

Global knowledge of other nodes naturally helps achieve a more efficient dissemination [11], but perfect global knowledge is impractical to achieve. However, strategies that learn as much as possible about other nodes in the system may benefit in the long run, as we show in §5. We are unaware of any prior work that account for both local and global data interests when disseminating.

## 3. RELEVANCE-DRIVEN DATA DISSEMINATION

We now introduce the basic concept of relevance-driven data dissemination, which uses *relevance scores* to decide whether to select items for transfer and which order to send them. The process delivers data items to nodes based on their interests rather than their network identifiers (like traditional networking).

To compute a specific node's relevance score for a data item, we assume that each data item carries metadata in the form of a set of *attributes* $\mathcal{D}$. Likewise, each node possesses a set of *interests* $\mathcal{I}$ that correspond to data attributes. Interests are forwarded to other nodes through direct contacts or through relaying them via third parties. A node's interests are weighted to emphasise particular interests of a node.

We classify data items as *relevant* to a node if the intersection $\mathcal{R} = \mathcal{D} \cap \mathcal{I}$ is non-empty. The strength of a node's interest in a data item is based on the matching interests $a \in \mathcal{R}$ and their respective weights $w_a$, expressed as a score:

$$score = \frac{\sum_{a \in \mathcal{R}} w_a}{\sum_{a \in \mathcal{I}} w_a} \quad (1)$$

The relevance score can be used by a dissemination system to determine the ordering in which items to be forwarded should be transferred. For example, Figure 1 shows a transfer schedule from node $j$ to node $i$, based on $i$'s relevance score of the data items carried by $j$. A high relevance score implies $j$ should transfer the item early in a contact to ensure its successfull delivery. Note that this relevance-driven ordering does not determine which encountered nodes are good "data mules", this is the task of a forwarding algorithm.

Although the above example represents a relatively straightforward approach to ordering that locally optimises for the nodes involved in the exchange, other more collaborative approaches, such as exchanging items also relevant to other nodes, may prove more beneficial for the network as a whole and in the long run. Ensuring a data item is maximally relevant to a receiver is an obvious greedy approach for increasing local interest satisfaction. However, in many cases, when the global interest in a data item is strong—although the local interest is weak—it may be preferable to prioritise that item's dissemination over one that maximises the local interest satisfaction. To study these various trade-offs in data dissemination, we define the following strategies for exchanging data items during a node contact:

**ScoreLocal** — Only locally relevant data items are exchanged, ordered by their relevance scores relative to the receiver of a data transfer, with the most relevant data item first.

**ScoreGlobal** — Only globally relevant data items are exchanged, ordered by their aggregate relevance scores relative to all nodes known at the time the data transfer occurs. The purpose of this strategy is to promote the dissemination of globally relevant items over locally relevant ones.

**RandomLocal** — Only locally relevant data items are exchanged, but they are ordered randomly. This is a baseline for evaluating the value of using the relevance score to order data items.

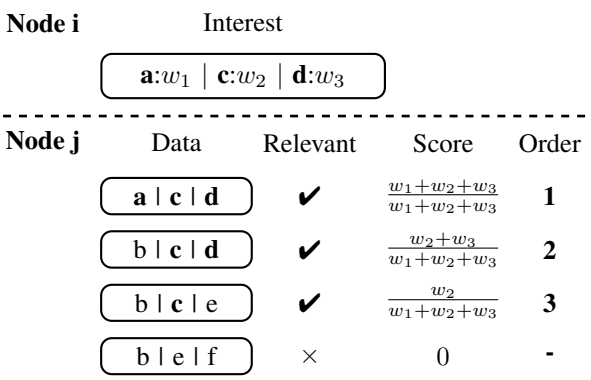



Figure 1: Data ordering according to a node's interests. The recipient $i$ has weighted interests causing the depicted ordering of $j$'s data items.

**Random** — Any buffered data items are exchanged in random order, irrespective of relevance to the nodes involved in the data exchange. This strategy provides a baseline for evaluating the value of using the relevance score to both *select* and *order* data items.

**LiFoLocal** — Only locally relevant data items are exchanged, ordered based on buffer time with the "freshest" item first. The purpose of this strategy is to promote fresh data, a property that has been shown to be efficient in buffer management in sparsely connected networks [3].

We shall refer to ScoreLocal, LiFoLocal and RandomLocal as "local strategies" as they select items for transfer based only upon the recipient's interests.

# 4. EXPERIMENTAL METHODOLOGY AND SETUP

In this section we describe the experimental methodology and setup we use to compare dissemination strategies. The strategies are implemented in Haggle [1] and we evaluate them through a set of experiments executed on a trace-driven emulation testbed, where contact traces decide the connectivity between Virtual Machine (VM) nodes [2]. We use three traces with different properties to minimise the bias on our results due to the configuration of a particular trace (§4.1), and understand transfer ordering across a range of network densities and churn. We distribute interests and data according to a well-known model of data and interests on the web (§4.2). A relevance metric from the area of information retrieval allows us to evaluate the relevance of the data delivered to nodes in our experiments (§4.4).

## 4.1 Contact Traces

We use three contact traces, generated either from topology/mobility models or real measurements, which each aim to subject our dissemination strategies to different network topologies:

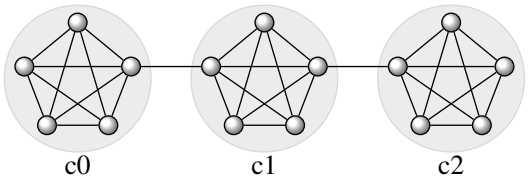



Figure 2: Clusters in Line topology, with three $K_5$ graphs connected by bridge links. Each edge is an intermittently connected Markov link.

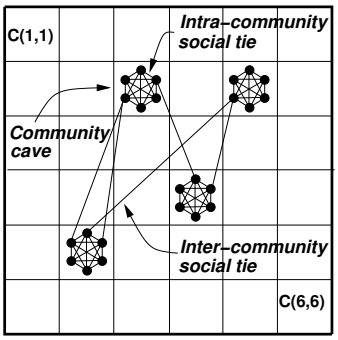



Figure 3: HCMM:SO contact model example: Each node is assigned a home cell and moves to other cells according to its social relations there (source [8]).

**Clusters in Line** — This is a synthetically generated trace that configures nodes in clusters with the goal of studying dissemination in a segmented network, as shown in Figure 2. The links in-between nodes in each cluster, and links between bridging nodes, are modeled by a two state Markov link model [10], with an expected contact duration of 60 seconds and inter-contact time of 240 seconds. Each of the three clusters comprises five nodes.

**HCMM:SO** — This trace is generated using the Home-cell Community-based Mobility Model with Social Overlay [8], which aims to accurately model both intra- and inter-community contact patterns. The intra-community pattern is achieved by assigning nodes to home cells (or caves) according to a caveman social graph [19], as illustrated in Figure 3. Nodes then move to another cell $C$ with a probability proportional to the number of "friends" (in the social graph) that have $C$ as their home cell. The inter-community contact pattern is modelled through periods of social activity, in which *community bridge nodes* convene at common locations according to a complementary social graph overlay. This model achieves weak non-homogenous mixing. We use a HCMM configuration with 30 nodes, in 6 communities a 5 nodes distributed over 10x10 cells, and a social overlay with edge activity of 2 hours but otherwise the same parameters as in [8].

**Real World** — A real-world trace, collected from an experiment with 10 mobile phones, complements our two synthetic traces. The mobile phones were handed out to col-

| Parameter | Symbol | Default Size |
|---|---|---|
| Attribute pool | $\mathcal{A}$ | 100 |
| Data attributes | $\mathcal{D}$ | 5 (Zipf, $\alpha = 0.5$) |
| Interest attributes | $\mathcal{I}$ | 5 (Zipf, $\alpha = 0.5$) |
| Data items per node | | 200 |

Table 1: Parameters used when generating (meta)data and interests.

leagues in our office corridors, who carried them during their work day. The phones used WiFi to send beacons every five seconds, allowing them to detect each other when in range. The WiFi interfaces used a low transmit power of 2 dBm to reduce battery consumption. The experiment lasted around six hours and people attended their normal work schedules; they sat in their offices, chatted with colleagues, had lunch, and participated in meetings.

## 4.2 Interest modeling and Data Distribution

To model a data-centric network, interests and data should be distributed in a way that reflects the skewness of item popularity in real life. Studies on user interest distribution have been shown to exhibit Zipf-like behaviour [6], and we therefore model our distribution of interests and content metadata accordingly. To this end, we form node communities by assigning every node a set of five attributes $\mathcal{I}$ from an attribute pool $\mathcal{A}$ of size 100. The attributes in $\mathcal{A}$ are chosen according to a Zipfian distribution with skewness $\alpha$. A higher skewness leads to a higher overlap of the interest between the nodes, while $\alpha = 0$ corresponds to a uniform distribution of the interests. This method allows to control the strength of the interest segregation among nodes. Each attribute $a \in \mathcal{I}$ is also given a weight $w_a$, proportional to the likelihood of picking $a$ from $\mathcal{A}$, and normalised such that the sum of a node's interest weights equals 100.

We assign each node data items, each with a set of five attributes $\mathcal{D}$ from the attribute pool $\mathcal{A}$ with the same probability distribution as above. We chose to inject all data items at the beginning of the experiments to enable complete analysis of a single wave of item distribution and so that each data item has equal conditions (i.e., node contacts) to disseminate.

After having explored different choices of $\alpha$ for the attribute pool $\mathcal{A}$, we settled on $\alpha = 0.5$ as default. This reflects a balanced attribute overlap, avoiding a situation where all nodes have interest in every data item. Instead, with our configuration, a node has an interest in $\sim 22\%$ of all data items. The experiment parameters are summarised in Table 1.

## 4.3 Data Transfer

To focus on the transfer ordering effect when forwarding, rather than *whether* an item should be forwarded, we artificially limit the number of transfers per node contact. This to emphasise the impact of limited contact duration, and to abstract from Bluetooth or WiFi bandwidth offered by mobile phones, data size, and the distribution of contact duration in the given contact traces.

The results presented in the following section are derived with a limit of 10 data objects per node contact. Different configurations with up to 100 data objects per node contact showed comparable relations between the different strategies, we therefore concentrate the analysis on one configuration. Furthermore, a low number of transfers reflects resource conservative nodes—a likely situation in real life.

## 4.4 Evaluation Metrics with Relevance Focus

To measure the effectiveness of relevance-driven dissemination we use the normalised Discounted Cumulative Gain (nDCG) [12], an established metric in the information retrieval community. The nDCG assigns each result set a quality valuation between 0 and 1, and accounts for the ordering of data items in the set of retrieved items in comparison with an ideal order. In our case, the ideal ordering is obtained for a node by taking the top $T$ sorted (according to relevance) data items out of all possible items, while the experienced ordering is the top $T$ sorted sequence of items that have been delivered to the node in the experiment. The nDCG metric thus gives a measure of how successful a strategy is at delivering data of high relevance. The nDCG is computed as follows:

$$nDCG^n[T] = n_T(s_1 + \sum_{i=2}^{T} s_i/log_b(i)) \qquad (2)$$

where $s_i$ is the score of the $i^{th}$ relevant received data item (*i.e.*, its score is $i$) and $1/n_T$ is the $nDCG^{n-ref}[T]$ for the relevant data available in the system used for normalisation. This measure will only consider how well the system provides a node's most desired data items, items below rank $T$ will not contribute utility to the node. The network-wide nDCG is calculated as the mean of the per-node nDCG$^n$, and gives a measure of a strategy's performance for the network as a whole. It can be seen that all nodes are weighted equally to this systemic measure, that is, nodes with low interest in available items contribute equally to the network-wide score.

Naturally, nDCG does not give a complete picture of the performance of a strategy, revealing nothing about the cost to achieve a certain nDCG or whether less relevant data is unnecessarily delayed in order to achieve a high nDCG. We therefore complement our analysis with traditional metrics, such as *delivery ratio*, *delay* and *overhead*. However, these metrics are well understood and require no further introduction.

# 5. RESULTS

In this section we look at the ability of dissemination strategies to deliver relevant items and how this evolves through the experiment. The time and delay aspect is important to understand the learning process of globally oriented strategies, which need time to learn more about the interests of

| Zipf $\alpha$ | 0.25 | | | 0.5 | | | 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|
| dst\src | c0 | c1 | c2 | c0 | c1 | c2 | c0 | c1 | c2 |
| c0 | 100 | 13 | 0 | 100 | 19 | 1 | 100 | 30 | 17 |
| c1 | 15 | 100 | 17 | 14 | 100 | 24 | 23 | 100 | 64 |
| c2 | 0 | 16 | 100 | 0 | 20 | 100 | 13 | 65 | 100 |

Table 2: Fraction of inter-cluster data delivered for different Zipf $\alpha$ in the *Clusters in Line* topology, using any local strategy.

other nodes. The cost of dissemination (*i.e.*, overhead) also reveals important trade-offs between performance and transfer efficiency.

## 5.1 Delivery ratio

Each node is interested in a subset of the data distributed across the nodes in the network. A single node's delivery ratio is the fraction of relevant items that have been received by the node. The network-wide delivery ratio is the mean node-specific delivery ratio. It gives an indication of how successful the network has been at delivering all data items of relevance to all nodes.

*Dissemination Across Segmented Networks.*

The distribution of data items and interests, the contact patterns, and the length of the experiment all constrain the delivery process. In particular, interest and data distributions affect strategies that only make local considerations. These strategies suffer when networks are segmented, because nodes isolated from each other must rely on other nodes to transfer the data they desire from other parts of the network. With local strategies, such transfers only occur when the relevant data items are also of interest to the recipient.

The *Clusters in Line* topology helps us to study the effect of segmented networks on local strategies. By changing the $\alpha$ parameter of the Zipf distribution, we can study how the node interests affect the dissemination across clusters. Table 2 shows the fraction of items delivered from one cluster to another. About 15% of relevant data items are able to traverse to adjacent clusters when $\alpha = 0.25$. No items traverse between the two non-adjacent clusters (c0 and c2). Increasing $\alpha$ to $0.5$ allows one percent of the items to traverse from c0 to c2, also slightly improving adjacent traversal. Raising $\alpha$ to $0.75$ improves the non-adjacent dissemination further. The delivery ratio across clusters is bound by the interests of the bridge nodes; only data of interest for the bridge node in cluster c0 will be disseminated from cluster c1 to cluster c0, for example.

As local strategies only transfer locally relevant data, every transferred item contributes to the delivery ratio independent of the ordering strategy. While the strategies eventually achieve the same delivery ratio, their progress is different as the availability of relevant data items becomes scarce. The next section discusses this aspect.

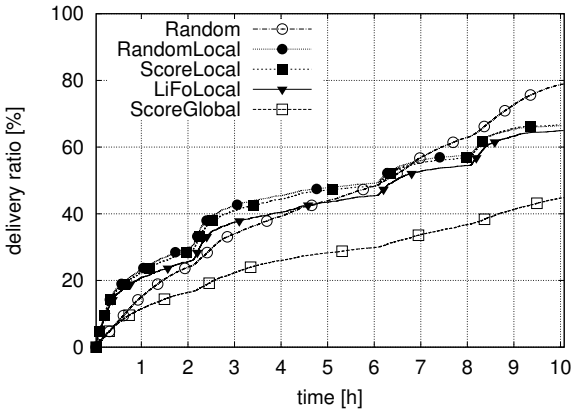



Figure 4: HCMM:SO: Delivery ratio progress over time, $\alpha = 0.5$.

*Delivery Ratio Progress.*

The rate at which relevant items are delivered gives an indication of the strategies' ability to prioritise the network-wide delivery ratio. HCMM:SO periodically exposes nodes to new neighbours, allowing us to examine the effect of the local saturation of neighbours' items. We can see this in Figure 4, where the progress of delivery ratio is given for all strategies. Local strategies perform well initially but start to experience limited availability of items left to share, reducing progress until new nodes are encountered (as seen at hours 2, 6, and 8). This pattern is repeated throughout the experiment. Random and ScoreGlobal strategies select more data items, and therefore suffer less from this local saturation effect.

Successful delivery of new items will be constrained when all contacts are used to distribute the same data, the network needs variety to ensure all interests are satisfied. This can be seen in the slow growth of ScoreGlobal, which aims to have all nodes swapping the same "best" items, starving delivery of less globally relevant items. For this reason, Random begins to outperform the others after 7 hours, as it enforces variety in dissemination among nodes.

Variety in data among the nodes is in particularly important when local saturation affects the ability to progress in data delivery. LiFoLocal performs worst of the Local strategies in terms of delivery ratio because promoting fresh data does not improve variety in data among the nodes the same way as random or score based ordering do.

## 5.2 Received Item Relevance

The delivery ratio by itself does not give any information about the ability of a strategy to disseminate the *most* relevant data. We measure the relevance of the data received by nodes compared to the most relevant data available in the system using the nDCG metric.

Figure 5 shows the network-wide nDCG progress evaluated over the 10 most relevant received data items for the

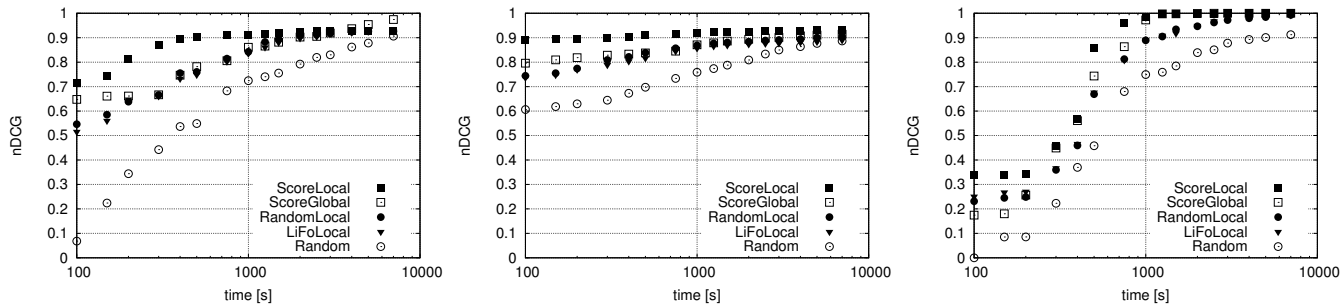



Figure 5: Normalized Discounted Cumulative Gain nDCG evaluated over the 10 most relevant received data over time of the experiment for (a) Clusters in Line (b) HCMM:SO (c) Real World

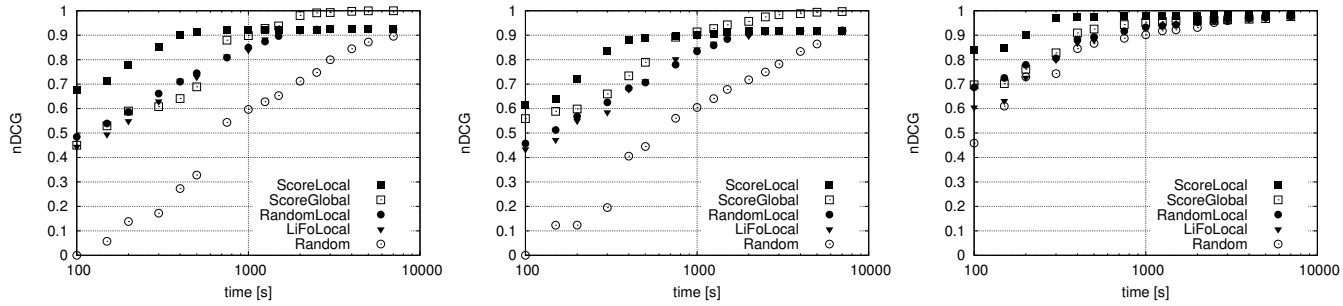



Figure 6: Normalized Discounted Cumulative Gain nDCG on *Clusters in Line* topology evaluated over the 10 most relevant received data over time of the experiment for varying values of $\alpha$ (a) $\alpha = 0$ (b) $\alpha = 0.25$ (c) $\alpha = 1$

three topologies. At the beginning of all experiments ScoreLocal consistently delivers the most relevant data, as the strategy was designed to exchange the most relevant data first. In the *Clusters in Line* topology we see the ScoreGlobal strategy eventually exceed the performance of ScoreLocal due to network segmentation from which local strategies suffer. It is also interesting to note that ScoreGlobal develops its knowledge about other nodes over time, at the first node contact it is equivalent to ScoreLocal, as it does not have previous knowledge about other nodes. Thus the strategy initially achieves a higher nDCG than random or LIFO based strategies but loses this behaviour over time as it meets more nodes and develops a global view. The HCMM:SO results begin high due to the large amount of early contacts.

In terms of relevance, using data freshness as an ordering criteria does not give a advantage over random ordering, indicating that the promotion of newly received does not have the same beneficial effect as it does in buffer management [3]. This suggests that relevance may have to be explicitly considered in any selection scheme that would like to deliver relevant data.

*Interest segregation.*

The effect of varying $\alpha$ for the *Clusters in Line* topology is depicted in Figure 6. If the $\alpha$ value is very high, attributes will be very skewed toward the top attributes. Thus, most nodes and items will share the highly likely attributes. When $\alpha$ is very low, attributes will be evenly distributed and so it will be unlikely that two adjacent nodes find the same item relevant. This lack of shared relevance limits items' ability to spread with a local strategy. We can see the effect of this in Figure 6, where the local strategies do not reach 100%. This is due to the segregation of nodes with a matching interest in items. Whereas non-local strategies can overcome these limitations, as transferred items do not need to interest the recipient.

With sparse topologies the path length to other nodes will be longer than in dense topologies. Longer paths are less likely for all adjacent nodes to share interest in data items. Thus shorter path lengths will not suffer from as much segregation.

As ScoreGlobal develops a more informed view of other nodes' interests, it will gain a more uniform view of interest distribution. A lack of strong views on what should be shared avoids segregation.

Informed ordering of data items is of particular concern when nodes have particularly segregated interests (low $\alpha$). Whereas with highly skewed interests (high $\alpha$) the Random strategy does not suffer from its simplicity.

*Relevance versus Delivery Ratio.*

Comparing relevance against the delivery ratio gives an

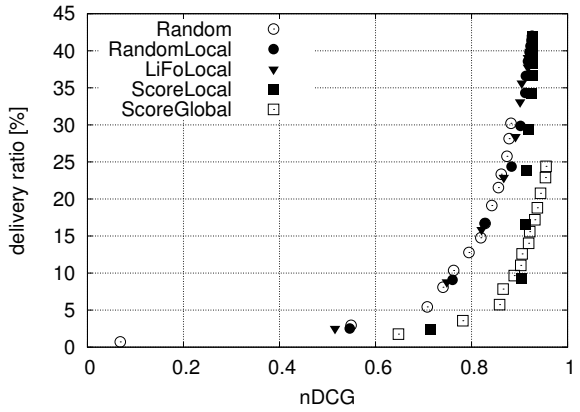



Figure 7: Clusters in Line: Normalized Discounted Cumulative Gain nDCG versus delivery ratio.

indication about the efficiency of the strategies. While, achieving a high nDCG with low delivery ratio is likely to involve transfer of irrelevant data, it shows that the few relevant data delivered actually is most relevant.

In Figure 7, we plot the nDCG against the delivery ratio. Each data point represents periodic instances in time as both delivery ratio and nDCG monotonically increase over the experiment. More separation between points thus shows better progress towards delivery and relevance.

Random and freshness based ordering roughly show the same behaviour, while the Random strategy has slowest progress. Relevance based ordering naturally achieves higher nDCG for the same delivery ratio than other orderings, with the global selection strategy ScoreGlobal overcoming the high relevance / low delivery property slower. This indicates the high cost of achieving its superior relevance score at the end of the experiment. ScoreLocal on the other hand cannot take advantage of the increasing delivery ratio to improve its nDCG score due to the previously discussed interest segregation.

## 5.3 Transfer Efficiency

We use the overhead in the number of transferred data items per received relevant data item as measure of efficiency. Table 3 summarises the overhead for different data an interest distributions for the *Clusters in Line* topology. While the overlap in data and interest distribution affects the overhead needed to disseminate the data, we found that the results are similar for the three topologies. Local strategies achieve optimal overhead of 1.0 as every transferred data item is relevant to its recipient.

Both ScoreGlobal and Random try to satisfy system benefit and therefore use contact opportunities to transfer data that is not relevant to its recipient. As the interests of the recipient are included in the aggregated interests used to select data, the chances of transferring relevant data items are higher for ScoreGlobal than using an entirely random strategy. Looking at the data and interest distribution, we observe that increased skewness leads to a reduction in overhead. This is because more nodes are interested in the same data item, increasing chances that a recipient happens to be interested in the item. The overhead of the ScoreGlobal strategy almost reaches optimality with a skewness of $\alpha = 1$, suggesting that the individual node interests are highly overlapping given that data and interest distribution.

| Zipf $\alpha$ | 0 | 0.25 | 0.5 | 0.75 | 1 |
|---|---|---|---|---|---|
| *Local | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| ScoreGlobal | 3.09 | 2.98 | 2.41 | 1.51 | 1.08 |
| Random | 3.77 | 3.63 | 2.97 | 1.86 | 1.24 |

Table 3: Clusters in Line: Overhead in terms of number of transferred data items per received relevant data item.

# 6. CONCLUSIONS

This paper proposed that the ordering of data items in transfer limited data-centric opportunistic networks has a large impact on the relevance of received data. We experimentally compared five strategies that select and order data to be transferred and evaluated their ability to deliver the most relevant data, measured using the normalised Discounted Cumulative Gain (nDCG). The trace-driven experiments on an emulation testbed included contact traces with a range of densities and mixing.

We saw that selection and ordering of transferred items significantly affects the achieved delivery ratio and its rate of increase, respectively. If nodes desire highly relevant data, careful ordering of limited transfers can satisfy these desires quicker than order agnostic approaches. With local strategies, if it is unlikely that neighbours are interested in the same content, segregation can occur where nodes are unable to disseminate items through the network. Interestingly, considering the relevance of data provides a way to overcome this segregation. In the case of densely connected networks, just local decisions can achieve system properties. However, in particularly sparse networks we need to explicitly consider non-local information to successfully promote system concerns.

We saw how selecting items to transfer using global information eventually outperforms local approaches. However, this does not mean that changing strategies in the experiment achieves the best of both approaches, as the early transfers dictate later performance.

We investigated limited transfer capabilities in data-centric opportunistic dissemination systems. We propose that the ordering of item transfers is a behaviour that, while rarely explicitly considered, has a large impact on the timely delivery of the most relevant data. For future work we plan to consider injecting data into nodes continuously, the correlation between node topology and interest structures and an analysis of item diversity and fairness aspects.

## Acknowledgements

This research was funded by the ResumeNet project under the EU grant FP7-224619. We thank Theus Hossmann for making the HCMM:SO code and Figure 3 available to us.

## 7. REFERENCES

[1] Haggle code project page, http://haggle.googlecode.com.

[2] BJUREFORS, F., GUNNINGBERG, P., AND ROHNER, C. Haggle testbed: a testbed for opportunistic networks. In *7th Swedish National Computer Networking Workshop* (2011).

[3] BJUREFORS, F., GUNNINGBERG, P., ROHNER, C., AND TAVAKOLI, S. Congestion avoidance in a data-centric opportunistic network. In *ACM SIGCOMM ICN Workshop* (2011).

[4] BOLDRINI, C., CONTI, M., IACOPINI, I., AND PASSARELLA, A. Hibop: a history based routing protocol for opportunistic networks. In *Proc. IEEE WoWMoM 2007* (2007).

[5] BOLDRINI, C., CONTI, M., AND PASSARELLA, A. Contentplace: Social-aware data dissemination in opportunistic networks. In *ACM MSWiM* (2008).

[6] BRESLAU, L., CUE, P., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM* (1999), pp. 126–134.

[7] HELGASON, O. R., YAVUZ, E. A., KOUYOUMDJIEVA, S. T., PAJEVIC, L., AND KARLSSON, G. A mobile peer-to-peer system for opportunistic content-centric networking. In *MobiHeld* (August 2010).

[8] HOSSMANN, T., SPYROPOULOS, T., AND LEGENDRE, F. Putting contacts into context: Mobility modeling beyond inter-contact times. In *MobiHoc* (Paris, France, May 2011).

[9] HUI, P., CROWCROFT, J., AND YONEKI, E. Bubble rap: Social-based forwarding in delay tolerant networks. In *MobiHoc* (2008), pp. 241–250.

[10] HWANG, S., AND KIM, D. Markov model of link connectivity in mobile ad hoc networks. *Telecommunication Systems 34* (2007), 51–58. 10.1007/s11235-006-9025-x.

[11] JAIN, S., FALL, K., AND PATRA, R. Routing in a delay tolerant network. *ACM SIGCOMM CCR 34*, 4 (2004).

[12] JÄRVELIN, K., AND KEKÄLÄINEN, J. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR* (2000).

[13] KRIFA, A., BARAKAT, C., AND SPYROPOULOS, T. MobiTrade: trading content in distruption tolerant networks. In *CHANTS* (September 2011).

[14] LENDERS, V., KARLSSON, G., AND MAY, M. Wireless ad hoc podcasting. In *IEEE SECON* (June 2007).

[15] LINDGREN, A., DORIA, A., AND SCHELÉN, O. Probabilistic routing in intermittently connected networks. *SIGMOBILE MCCR* (2003).

[16] LINDGREN, A., AND PHANSE, K. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Comsware* (2006).

[17] MOGHADAM, A., AND SCHULZRINNE, H. Interest-aware content distribution protocol for mobile disruption-tolerant networks. In *WoWMoM* (2009).

[18] MTIBAA, A., MAY, M., DIOT, C., AND AMMAR, M. PeopleRank: Social opportunistic forwarding. In *IEEE INFOCOM 2010* (2010).

[19] NEWMAN, M. E. J. The structure and function of complex networks. *SIAM REVIEW 45* (2003), 167–256.

[20] NORDSTRÖM, E., GUNNINGBERG, P., AND ROHNER, C. A search-based network architecture for mobile devices. Tech. Rep. 2009-003, Department of Information Technology, Uppsala University, Jan. 2009.

[21] SOLLAZZO, G., MUSOLESI, M., AND MASCOLO, C. TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. In *MobiOpp* (June 2007).

[22] SPYROPOULOS, T., PSOUNIS, K., AND RAGHAVENDRA, C. S. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *ACM SIGCOMM WDTN Workshop* (2005).

[23] VAHDAT, A., AND BECKER, D. Epidemic routing for partially-connected ad hoc networks. Tech. rep., 2000.