
Department of Information Engineering, University of Pisa, Italy

*Ph.D. Thesis - XIX Cycle*

# Computer-network Solutions for Pervasive Computing

Ph.D. Candidate

*Luciana Pelusi*

Advisors

*Prof. Giuseppe Anastasi*

*Dott. Marco Conti*

April 2007

*Valentina*

# Contents

# List of Figures

# List of Tables

# 1. Introduction

The last decades have witnessed a revolutionary change in the computing model with the advent of mobile wireless communications. In the nineties, mobile computing devices (e.g., laptops, personal digital assistants and other generic handheld digital devices like cellular phones), have finally met consensus of users and experienced widespread diffusion. Users have discovered the possibility to access all the information they require *whenever* and *wherever* needed and have interestingly changed their habits in using computers. While before nineties the use of computing devices was only concerned with job and study, and devices were intended to be powerful desktops located in offices, at home or even in Internet cafes were Internet access were provided, in the so-called *mobile computing age*, people have experienced, and enjoyed, new ways to take advantage from digital devices and have thus started to utilize different electronic platforms at the same time. Mobile access to the Internet has been one of the first incentives to the use of portable computing devices. Mobile users have started to utilize their cellular phones to check e-mails and browse the Internet, and travelers with portable computers have begun surfing the Internet from airports, railway stations, and other public locations.

From then on, ever more powerful mobile devices have come to the market thank to the rapid progresses in both hardware and software. These days portable computers do not only replicate but almost substitute desktop computers. But the most interesting evolution consists in the ongoing proliferation of even more smaller devices pervading the environment, from PDAs to wearable computers, and to devices for sensing and remote control. This current age has therefore been named *Ubiquitous Computing age* [Wei91], or *Pervasive Computing age* [Sat02]. These definitions refer to an environment which is saturated with computing and com-

munication capabilities aimed to help users in their everyday life but without requiring any major change in their behavior. Virtually everything in this environment (from key chains to computers, and PDA's) is connected into a network, and can originate and respond to appropriate communications. Similar environments are not actually deployed yet, but a physical world with pervasive, sensor-rich, network-interconnected devices embedded in the environment is envisaged as the next generation computing challenge [ECPS02].

The nature of ubiquitous devices makes wireless networks the easiest solution for interconnection. Furthermore, mobility is a fundamental part of everyday life. Hence, wireless and mobile communications are a fundamental building block for smart pervasive computing environments. Currently, wireless connectivity is generally provided by *wireless LANs* (conforming, for example to the IEEE 802.11 WLAN standard), or somewhat smaller and less expensive *Bluetooth* devices. However, in a pervasive computing environment, the infrastructure-based wireless communication model is often not adequate because it takes time to set up the network infrastructure and the costs associated with installing the infrastructure can be quite high. These costs and delays may not be acceptable for dynamic environments where people and/or vehicles need to be temporarily interconnected in areas without a pre-existing communication infrastructure (e.g., inter-vehicular and disaster-relief networks), or where the infrastructure cost is not justified (e.g., in-building networks, specific residential communities networks, etc.). In these cases, infrastructure-less or *ad hoc* networks provide a more efficient solution. The simplest ad hoc network is a peer-to-peer network formed by a set of stations within communication range of each other that dynamically configure themselves to set up a temporary single-hop ad hoc network. The widespread adoption of the Bluetooth technology in computing and consumer electronic devices makes the Bluetooth piconet the most relevant solution for single-hop ad hoc networks. However, a single-hop ad hoc network only interconnects devices that are within the same transmission range. This limitation can be overcome by exploiting the multi-hop ad hoc (MANET) technology. In a MANET, the users' mobile devices cooperatively provide the functionalities usually provided by the network infrastructure (e.g. routers, switches, servers). Devices that are not directly connected, communicate by forwarding their traffic

via a sequence of intermediate devices.

Sensor nodes are an important component of pervasive systems, and wireless ad hoc networking techniques also constitute the basis for sensor networks. However, the special constraints imposed by the unique characteristics of sensing devices, and by the application requirements, make many of the solutions designed for multi-hop wireless networks (generally) not suitable for sensor networks. For this reason, the ad hoc networking research community has recently generated an extensive literature dedicated to sensor networks.

This thesis goes through the above wireless scenarios, from the widely deployed *WLANs* to the emerging *Opportunistic Networks*. Some of the most relevant aspects of wireless pervasive systems are reviewd, starting from energy efficiency, going further to connectivity and reliability issues in transmissions.

In *infrastructured* wireless scenarios, like *WiFi hotspots*, the presence of base stations, specifically Access Points (APs), guarantees high connectivity and high bandwidth to the nodes in their proximity. However, only limited coverage is available and this solution seems to be suitable only for temporary user needs, or for users with limited mobility. Moreover, a WLAN is to be considered an extension of the wired Internet, in fact, scenarios with mobile wireless devices accessing the Internet via APs are generally referred to as *wireless Internet*. These environments provide the same identical kinds of services as the wired Internet. Hence, their major requirement is to guarantee the same service quality as the wired Internet. This implies masquerading the last wireless connection between the AP and the mobile device. A key issue in wireless Internet is *energy efficiency* since portable devices are battery-fed and have therefore limited energy budget. Though technological progress has been producing in the last few years ever more powerful batteries, the increased usage of portable devices has also led to higher energy requirements. Hence, software solutions are currently being deployed to avoid energy wastage at the wireless devices during their common activity. When integrating energy saving support into the classical Internet applications, service provisioning in the wireless Internet is possible for a longer time. The most efficient energy saving solutions are those which save energy during networking activities since they have been demonstrated to be the most energy-hungry activities. Chapter II focuses on

the diffusion of multimedia streaming services in smart environments and specifically investigates scenarios where mobile users who have a Wi-Fi access to the Internet receive audio files from remote streaming servers [ACG+06a] [ACG+05a]. When dealing with these applications, particular attention must be taken to quality of service(QoS) requirements, and the solutions for optimization of resource consumption must not degrade the QoS perceived by the user. Energy saving is addressed by including periodic transmission interruptions in the schedule of audio frames at the server, such that the network interface card at the receiver can be set to a low-power consuming state. Experimentation on a software prototype shows that the solution proposed makes possible energy savings ranging from $76\%$ to $91\%$ (depending on the data rate and the background traffic) of the total consumption due to the network interface. Good user level QoS is also preserved.

*Infrastructure-less* ad hoc networks have their strength in the absence of infrastructure which means that they can be set up very fast and with minimum effort and low cost (only that of the wireless network interface card). Much work on wireless ad hoc networks has been conducted so far, mostly through simulation modeling and theoretical analyses. Instead, relatively few experiences with real ad hoc networks have been performed, and the results from these experiences do not always validate simulative conclusions. So, analytical and simulative results have to be complemented by real experiences (e.g., measurements on real prototypes) which provide both a direct evaluation of networks and, at the same time, precious information to realistically model these systems and avoid unrealistic simulation settings. Chapter III presents and discusses some lessons learned from an experimental work on a wireless ad hoc testbed [BCDP04]. Specifically, results are presented for a fully functioning prototype implementing a p2p middleware (*FreePastry*) on top of a multihop ad hoc network based on IEEE $802.11b$ technology . Recently, for this technology, [GLNT] has pointed out the existence of an ad hoc horizon (2-3 hops and 10-20 nodes) after which the benefit of multihop ad hoc networking vanishes. All the experiments performed fall inside this ad hoc horizon. The aim of the experimentation has been to identify solutions for this realistic setting and to quantify the Quality of Service (QoS) that the system is able to provide to the users. The measurements performed have pointed out that also in this limited setting, several problems still exist to construct efficient

multi-hop ad hoc networks. Cross-layering seems to be an effective approach to fix some of the problems identified in this analysis.

The feasibility of an ad hoc network is related to the attitude of nodes to stay in group such that they can communicate directly with each other (over single-hop connections) or through intermediate nodes over multi-hop connections. In both cases, communications rely on a certain knowledge of the network topology, at least of the path to the destination node. So, when a node has to send data to a destination node, it collects information about the reachability of that node. Then it sends out the data only if the destination node is actually reachable, along a pre-determined path, otherwise it discards the data. This is exactly what happens in the wired Internet where a message is either forwarded immediately or considered undeliverable. Actually, in wireless environments with high mobility, the network topology is continuously varying and it is possible that messages that cannot be delivered at a certain time can be delivered a few time later instead. Novel communication paradigms currently under study do not rely on the presence of a complete path towards a destination node but rather exploit delivery probabilities towards that destination node, referring to the possibility that a path to that destination will appear sooner or later. Obviously, this new communication paradigms do not suit all kinds of applications, especially real-time applications. Applications that are relatively independent of the time and can tolerate even long transmission delays are rather concerned. Chapter IV focuses on this new, emerging research area of wireless networking, known as *Opportunistic Networking*. The very characteristic of opportunistic networks is their intermittent connectivity, meaning that there is no assurance to the existence of a complete path between pairs of nodes wishing to communicate. Therefore, communications typically happen by exploiting pair-wise contacts between nodes. A node wishing to send/forward a message looks for a next-hop. If a next-hop exists then the node forwards the message to it, otherwise it keeps the message until a new contact opportunity arises. A message gradually approaches the destination by following the motion of the nodes it is carried by and by passing node-by-node during pair-wise contacts. Eventually, it reaches the destination. *Contact opportunities* among nodes may be scheduled over time or completely accidental. They depend on the particular network scenario and the mobility of nodes. The very novelty of

opportunistic networks consists in the fact that while infrastructured WLANs and infrastructure-less MANETs work *despite* mobility of nodes and can *tolerate* it up to a certain degree, opportunistic ad hoc networks actually *rely* on node mobility because full connectivity is not assured in any way and the probability of successful message delivery is as much higher as the nodes move and meet up with each other. This communication paradigm is particularly suitable to environments with a high number of heterogeneous devices. Opportunistic network scenarios are very challenging due to the scarceness of resources and are often referred to as *extreme networks*. Chapter IV overviews the theoretical foundations of opportunistic ad hoc networks and also describes some recent experimental studies and projects in the area. Finally, it reviews the state-of-the-art of the routing and forwarding strategies used in opportunistic environnments [PPC06b] [PPC06a].

Data forwarding is surely one of the most compelling issues in opportunistic scenarios and it concerns, in a certain sense, with *guessing* the best direction(s) towards the destination node. However, this is only part of the problem, since we can more generally talk about *reliability issues* in opportunistic environments and classify them into two different levels: *one-hop* and *end-to-end* reliability. One-hop reliability concerns with transmissions that occur during a contact between nodes in the communication range of each other, whereas end-to-end reliability concerns with efficacy of the data forwarding strategy applied. One-hop reliability is affected by the duration of contact times occurring between pairs of nodes. In fact, depending on the trajectories followed by nodes and on their speed, contact times between nodes can be shorter or longer, but are generally unpredictable. The duration of contact times limits the total amount of data that can be transferred between the communicating nodes and thus the overall throughput that can be achieved. Moreover, to make things worse, it should be noted that contact times are not well-delimited. Rather, their starting point and ending point can be inferred from the loss pattern which is experienced during transmissions [ACG+06b]. Initially, when a sender node moves towards a destination node, transmissions are highly disturbed and unstable leading to severe data loss. Then, when gradually approaching, the distance between the communicating peers gradually shortens and transmissions become more successful up to the optimum. Finally, they turn to worsen again when the sender goes by. The contact

time is defined as the time interval in which data loss is lower than a minimum threshold. Data transmission schemes suitable to opportunistic networks should allow successful transmission of data at each pair-wise contact despite the nodes approaching or going far apart from each other. Retransmission-based schemes, in particular, should be avoided because time-consuming. End-to-end reliability instead is mainly affected by the forwarding strategy which is responsible for finding hop-by-hop a path to the destination. As is discussed in chapter IV, many forwarding strategies have been conceived for opportunistic networks in the last few years. The first attempts have principally been on *epidemic-based* solutions [VB00] that simply flood messages all over the network. Epidemic dissemination guarantees *short delays* but is extremely *resource consuming* both in terms of memory occupancy and bandwidth usage. Later on, new sophisticated policies have been introduced to select only *one* or *few* next-hop nodes for a message. The choice of a next hop is generally based on statistics on the delivery success that a node can guarantee in keeping *custody* of a message. The delivery success rate that a relay node (next hop) guarantees to transmissions towards a destination node can be worked out in many different ways based on past encounters or visits to particular places [BBL05] [LDS03], on specific context information [MHM05], etc.
Targeting both one-hop and end-to-end reliability, chapter V introduces special *Encoding techniques* which are very promising to opportunistic communications, namely *erasure codes* and *network coding* [PPC]. The main idea of encoding techniques applied to networking protocols is to avoid sending plain data, but rather combining (encoding) together blocks of data, in such a way that the coded blocks can be somehow interchangeable at the receivers. In the simplest example, a source node willing to send $k$ packets actually encodes the $k$ packets into $n$ encoded packets, with $n \gg k$. Encoding is performed such that a receiving node does not need to receive exactly the original $k$ packets. Instead, any set of $k$ encoded packets out of the $n$ encoded packets, which have been generated at the source, is sufficient to decode the original $k$ packets. Different receivers might get different sets of encoded packets, and will still be able to decode the same original information. Adding redundancy to transmissions is an important feature in all kinds of encoding techniques. It improves one-hop reliability since receivers are enabled to tolerate a certain degree of data loss: when data gets lost, the desti-

nation simply waits for other subsequent data to arrive and re-transmissions are generally limited to few critical cases. Eventually, the reconstruction of the original data is faster than when relying on re-transmissions only. As reguards end-to-end reliability, encoding techniques are well-suited to support data forwarding strategies that exploit both multi-path and multi-user diversities. Namely, the original volume of information is subdivided in many chunks which are encoded in groups such that for each group of information chunks a greater group of encoded chunks is generated with some redundancy added. Chunks of data can be spread out in different directions and exploit different paths. The benefits are twofold: on the one hand, the delivery probability increases since the higher is the number of paths attempted, the more probable it is that at least one successful path is found[1]. On the other hand, if a few successfull paths exist and are traversed in parallel, then the overall information experiences shorter *delay* than if it passes in sequence through the same path.

Focusing on one-hop reliability, chapter VI presents the case study of a *sparse Wireless Sensor Network (WSN)* where sensor nodes are scattered all over the network area and connectivity is achieved by means of a mobile sink, named dataMULE. The dataMULE moves around in the network area and gathers information from sensor nodes when passing by them. The chapter discusses preliminary results from a real experimentation on a small set of *mica2* sensor nodes. In the experiments, blocks of data originated at the sensor nodes are combined (encoded) together to generate new blocks that are sent out instead of the original ones. The number of coded blocks is higher than the number of original blocks and allows good tolerance against data corruption and packet loss and finally leads to dependable data collection at the dataMULE. The aim has been to contrast the transmission pattern resulting from adoption of encoding techniques with that resulting from a classical re-transmission based protocol. Results show that good performance improvements are guaranteed when applying erasure coding with respect to re-transmission-based techniques.

[1] A path is unsuccessful either when the nodes that belong to this path never meet the destination or when the information carried is lost.

# II. WiFi Hotspots and Real-time Streaming Applications: an Energy-efficient Solution

## 1. Introduction

The increasing diffusion of portable devices and the growing interest towards pervasive and mobile computing is encouraging the extension of popular Internet services to mobile users. However, due to the differences between mobile devices and desktop computers, a big effort is still needed to adapt the classical Internet applications and services to the mobile wireless world. Energy issues are perhaps the most challenging problems since portable devices typically have a limited energy budget [ACP05].

Among the set of potential mobile Internet services, multimedia *streaming* is expected to become very popular in the near future. Streaming is a distributed application that provides on-demand transmission of audio/video content from a server to a client over the Internet while allowing playback of the arriving stream chunks at the client. The playback process starts a few seconds (*initial delay*) after the client has received the first chunk of the requested stream, and proceeds in parallel with the arrival of subsequent chunks. Before playback, the arriving chunks are temporarily stored at a *client buffer*. The streaming service must guarantee a good playback quality. This imposes stringent *time constraints* to the transmission process because a timely delivery of packets is needed to ensure their availability at the client for playback. Moreover, since constant-quality

encoded audio/video content produces *Variable Bit Rate (VBR) streams*, matching the time constraints is a real challenge in the best-effort Internet environment where the network resources are highly variable. Hence, one of the main goals for a streaming server is to use a *transmission schedule* that tunes the outgoing traffic to the desired network and client resource usage, while also meeting the playback time constraints. A good scheduling algorithm should allow an efficient use of the available network resources without overflowing or under-flowing the client buffer. Moreover, it should limit the initial delay before playback, because users do not generally tolerate high initial delays (greater than $5$–$10$ s), especially when waiting for short sequences. Many smoothing algorithms have been proposed in the literature [FR97]. However, none of them can be considered the best one because the performance metrics to be considered are often conflicting and cannot be optimised all together. Moreover, the same algorithm generally performs differently when applied to different media streams with different characteristics (e.g., *burstiness*). As a result, the most suitable smoothing algorithm may change every time depending on (i) the content of the media stream to be sent, (ii) the specific performance goals of the system components, (iii) the particular resource-usage policy adopted at the server, the client and at the network routers, respectively.

In this chapter we focus on streaming services in mobile wireless scenarios where servers are supposed to be fixed hosts in the Internet, whereas clients are supposed to run on mobile devices (e.g., PDAs, laptops or mobile phones with wireless network interfaces). Client devices access the Internet through an intermediate access point (or base station). We will assume the presence of a *proxy* between the client and the server since proxies are commonly used to boost the performance of wireless networks. In this environment we will concentrate on energy efficiency and will propose a proxy-based architecture based on the *Real Time and Power Saving (RT_PS)* protocol. This protocol implements an energy management policy whose main goal is to minimize the energy consumed by the *Wireless Network Interface Card (WNIC)* at the mobile host, while guaranteeing the real-time constraints of the streaming application. In this chapter we will refer to the Wi-Fi technology [WSf]. However, our solution is flexible enough to be used with any type of infrastructure-based wireless LAN. Our approach allows the WNIC to save a significant amount (up to $91\%$) of the energy typically consumed without energy

management. Another challenging issue, in this area, is to minimize the client buffer size, as memory costs for some kinds of mobile devices are still high. We will show that our solution achieves good energy savings even with small buffer sizes (e.g., 50 KB).

The remainder of the chapter is organized as follows. Section 2 describes some related work on energy management with emphasis on those pieces that refer to real-time streaming applications. Section 3 introduces the proxy-based architecture that we propose, while Section 4 describes the *RT_PS* protocol. Section 5 and Section 6 are devoted to the description of some specific aspects of the *RT_PS* protocol. Section 7 presents an experimental analysis of our proposal carried out on a prototype implementation, and finally, Section 8 concludes the chapter.

## 2. Related Work

Energy issues in wireless networks have been addressed in many papers. [ACL00] [KK98] [ANF05] found that one of the most energy-consuming components in mobile devices is the WNIC since networking activities cause up to 50% of the total energy consumption (up to 10% in laptops, up to 50% in hand-held devices).

A first approach to reduce the energy consumption of the WNIC is to improve the throughput at the transport layer. The work in [BB97] proposed the *Indirect-TCP* architecture as an alternative to the classical TCP/IP architecture for communications between mobile and fixed hosts. The Indirect-TCP *splits* the TCP connection between the mobile host and the fixed host in two parts, one between the mobile host and the Access Point, the other between the Access Point and the fixed host. At the Access Point a running daemon is in charge of relaying data between the two connections. The Indirect-TCP avoids the typical throughput degradation caused by the combined action of packet loss in the wireless link and the TCP congestion control mechanism. As a result, the total transfer delay successfully reduces.

Nevertheless, the reduction in the total transfer delay can only produce a slight energy saving if compared to the energy wastage caused by the inactivity periods of the WNIC during data transfers. In fact, the natural *burstiness* of common

Internet-traffic patterns determines long idle periods for the WNIC, when it neither receives nor transmits, but drains a large amount of energy the same, as is shown in [ACL00] [ANF05] [SK96]. A more effective solution to significantly reduce the energy consumption is to switch the WNIC *off* (or putting it in the *sleep mode*) when there is no data to send/receive.

The work in [ANF05] retains the standard TCP/IP architecture, and either uses the 802.11 Power Saving Mode (PSM), or keeps the WNIC continuously active, based on predictions about the application behavior in the near future. [ACL00] [KK98] [ACGP03b] [ACGP03a] use the Indirect-TCP architecture and also switch off the WNIC during the idle periods. [ACGP03b] proposes an application-dependent approach tailored to web applications and exploits the knowledge of statistical models for web traffic to *predict the arrival times* of the traffic bursts. [ACGP03a] proposes an application-independent approach, which is more suitable when several applications are running concurrently on the same portable device. Works in [ACGP03b] [ACGP03a] also highlight how traffic burstiness can be *exploited* to save energy. In fact, since switching the WNIC between different operating modes has a cost, having *few long* idle times rather than *several short* idle times is preferable from an energy saving standpoint.

The solutions so far mentioned are not suitable for streaming applications because of the time constraints imposed by such applications to the transmission process. Existing energy-management solutions for real-time applications are based on both the two following general approaches: (i) switching the WNIC to the sleep mode during inactivity periods, and (ii) reducing the total transfer delay of the streaming session. Switching the WNIC completely off is not typically considered because too time-consuming and thus potentially dangerous for the fulfilment of the real-time constraints.

*Transcoding techniques* have been introduced to shorten the size of audio/video streams so as to reduce the transfer delay [AS98] [BC00]. However, while leading to little energy savings (see above), these techniques may determine significant reduction in the *stream quality*.

The IEEE 802.11 PSM is not suitable for streaming applications because, as is stated in [CV02], it only performs well when the arriving traffic is regular, whereas

it is not suitable for popular multimedia streams. All the solutions described below therefore assume that PSM in not active.

Some application layer techniques have been introduced that rely on predictions on the arriving traffic behavior. They switch the WNIC to the active mode when a packet is expected to arrive and to the sleep mode when an idle period is expected to start. [Cha03] and [SR03] propose solutions with client-side predictions of the packet inter-arrival times based on the past history. Wrong predictions result in packet loss and quality degradation. Techniques based on client predictions can lead to energy savings ranging between $50\%$ and $80\%$ [Cha03] however, similarly to PSM, the best performance is achieved when the incoming traffic is regular. Predictions can thus benefit from some traffic shaping to become more effective. However, traffic shaping at the server is useless because transmission over the Internet changes the traffic profile by introducing *jitter* in the delay experienced by the packets. Therefore, traffic shaping must be performed at a proxy close to the mobile host. The solutions proposed in [CV02] and [SR03] perform traffic shaping at the proxy and also make use of *client-side predictions* to switch the WNIC to the sleep mode during the idle periods. The work in [SR03] also proposes a second solution where predictions are *proxy-assisted*. The proxy sends to the client a sequence of consecutive packets in a single burst. Then, it informs the client that the data transmission will be suspended for a given time interval. The client can thus switch the WNIC to the sleep mode for the announced time interval. The total energy saving achieved in [SR03] ranges from $65\%$ to $85\%$ when using client-side predictions, and from $80\%$ to $98\%$ with proxy-assisted predictions. These results have been obtained by reducing the energy consumption of both the WNIC (which is put in the sleep mode during inactivity times), and the CPU (whose usage is decreased thanks to transcoding).

Our solution relies on a proxy-based architecture where the proxy uses an *ON/OFF schedule* for data transmissions to the clients, i.e., it alternates transmission and non-transmission periods. In addition, the proxy warns the client as soon as a non-transmission period is starting so as the client can accordingly switch the WNIC to the sleep mode. In our proposal, unlike the solution in [SR03], the streaming policies are decoupled on the wireless and in the wired networks. In addition, the transmission schedule is dynamically adjusted by the proxy based on the current

available bandwidth as well as on the client buffer level.

We focus on *Stored Audio Streaming* applications, and mainly refer to MP3 audio files. Moreover, we have decided not to use transcoding techniques to preserve the audio quality since the MP3 compression already reduces the quality of the audio stream. We believe that only video streams can take advantage of transcoding techniques because they allow a significant reduction in the file size without severely affecting the playback quality [SR03].

## 3. Proxy-based Architecture

The proxy-based architecture that we refer to in our solution is depicted in Fig. II.1. The proxy functionality is supposed to be implemented at the Access Point. Therefore, the data path between the server and the mobile client is split at the border between the wireless and the wired networks (like in the Indirect-TCP model). The target in each part of the data path is different. In the proxy-client communications the major goal is to reduce the energy consumption at the mobile host. On the other hand, for the proxy-server communications a classical, smothness-oriented approach to the streaming is required. Smoothing is a good approach for the wired network since, by reducing the peak transmission rate, it improves the network-resource usage. However, it also increases the total transfer time and reduces the traffic burstiness, thus letting less room for energy management policies.

The streaming server is located at the fixed host and sends streams to the proxy (Access Point). Its scheduling policy is smoothness-driven. Real-time data is encapsulated in *RTP* (*Real Time Protocol*) [SCFJ03] [Sch99] packets while the *Real Time Streaming Protocol (RTSP)* [SRL04] is used to exchange control information for playback with the other streaming peer.

Since RTP uses UDP as underlying transport protocol, and UDP provides no built-in congestion control mechanism, the presence of multimedia traffic in the Internet can potentially lead to congestion. The *TCP-Friendly Rate Control* (*TFRC*) [HFPW03] protocol adaptively establishes an upper bound to the server transmis-

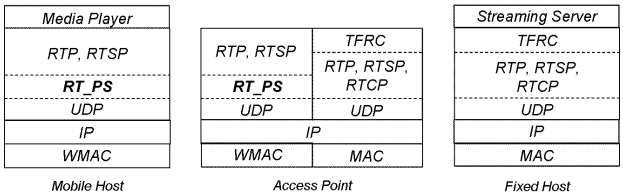


Figure II.1.: Streaming architecture for a wireless scenario.

sion rate, thus leading to congestion avoidance. RTP and *RTCP* (*Real Time Control Protocol*) [Hui03] packets are used in order to carry TFRC information [Gha04].

The *Real Time and Power Saving* (*RT_PS*) protocol, which is the core of this architecture, is located on top of the UDP transport protocol and implements the energy management policy. Since the focus of this chapter is on energy efficiency, hereafter we will only focus on the proxy-client communications over the wireless link. The *RT_PS* protocol works as follows. The proxy schedules packet transmissions by following an *ON/OFF scheme*. During the *ON periods* the proxy transmits packets to the mobile host at the highest possible rate allowed by the wireless link. During the *OFF periods* the mobile host switches the WNIC to the sleep mode. The traffic shaping performed by the proxy relies on the *knowledge of the audio/video frame lengths*, the client *buffer size*, and the current *available bandwidth* on the wireless link. During an ON period the proxy decides whether to stop transmitting or not depending on the available bandwidth. When the available bandwidth is high, the proxy continues transmitting until the client buffer fills up. When the available bandwidth is low, it stops transmitting to avoid increasing congestion and transfer delay. The duration of an OFF period is decided by the proxy when it stops transmitting, and depends on the client buffer level. An OFF period ends when the client buffer level falls down a dynamic threshold, named *low water* level, which warns the proxy about the risk of playback starvation. Finally, to avoid bandwidth wastage, only the frames that are expected to arrive in time for their playback are delivered to the client whereas the others are *discarded*. The computation of the frame arrival times relies on the estimates of the available throughput on the wireless link.

In addition to the client buffer, the energy management policy implemented by

the *RT_PS* protocol also relies on the *initial playback delay*. When the streaming session is established the playback process does not start immediately, instead, a given amount of data is accumulated in the client buffer before the playback process can start. However, it is worth noticing that the client buffer and the initial playback delay are not special requirements of our protocol. Indeed, they are needed in any streaming system to absorb the delay jitter introduced by the network.

# 4. RT_PS Protocol Description

The *RT_PS* protocol includes a proxy component running at the proxy and a client component running at the mobile device. The proxy-side component carries out the followings tasks: i) evaluates the playback *start point*, i.e., the initial playback delay, ii) transmits the (audio) frames to the client during the ON periods, iii) keeps track of the level of the client buffer by considering the progressions of both the transmission and the playback processes, iv) decides whether or not to stop transmitting given the current available throughput, and v) calculates the duration of the sleep periods for the WNIC of the mobile device in such way to prevent the playback process from starving due to buffer underflow.

The client-side component performs the following tasks: i) sends the initial request for an (audio) file to the proxy also specifying the size of the client buffer, ii) triggers the playback process when the start point elapses, iii) collects the arriving (audio) frames in a buffer where the media player can take and play them out, iv) estimates the throughput currently available on the wireless channel and notifies the *RT_PS* proxy-side component, v) switches the WNIC to the sleep mode upon receipt of a sleep command from the proxy, and vi) switches the WNIC back to the active mode as soon as the sleep time has elapsed.

A complete description of the *RT_PS* protocol is omitted for the lack of space (the reader can refer to [ACG+05b]). However, some key tasks are detailed in Section 5 and Section 6.

# 5. Start Point and Sleep Time Computations

This section describes how the *RT_PS* protocol derives the start point and the durations of the ON and OFF periods. It makes use of the following analytical model to refer to the streaming process over time.

## 5.1. Analytical Model

Fig. II.2 represents the playback process of an MP3 audio file by means of a graphical model [FR97] [Zha96] (symbols are described in Table II.1). An MP3 audio file is a sequence of frames and its playback begins at the *start point* ($SP$ in the figure), which is the time when the first frame of the file is played out. Each subsequent frame must be played out starting at fixed, equally spaced, times called *playback times*. In Fig. II.2 frames are spaced by $\Delta$ seconds.

The *playback function* $V(t)$ defines the total number of bytes that *must have been* played out by time $t$ in order to correctly reproduce the stream at the client. Specifically, $V(t)$ is defined as follows:

$$V(t) = \begin{cases} 0 & if \;\; t < SP \\ \sum_{i=0}^{k} f_i & if \;\; t \in [SP + k\Delta,\; SP + (k+1)\Delta) \end{cases} \qquad \text{(II.1)}$$

$k = 0,\ \ldots,\ N-1$; $i = 0,\ \ldots,\ k$. $f_i$ is the size in bytes of the $i^{th}$ frame in the stream, and $k$ is the number of the $\Delta$-*sized* inter-frame spaces occurred from the starting point $SP$ up to time $t$. The playback times can thus be formally defined as $SP + k\Delta,\ k = 0,\ \ldots,\ N-1$, and are actually deadlines for frame arrivals at the client, and for frame deliveries at the proxy. During a streaming session, frames usually arrive at the client ahead of their designated playback times, and are thus temporarily stored in the client buffer, whose size will be hereafter referred to as $B$. As $V(t)$ represents a lower bound for the number of bytes arrived at the client by time $t$, it is possible to define an *upper bound* for this number of bytes, denoted as $V^B(t)$. $V^B(t)$ is the maximum number of bytes that the client can host by time $t$ without causing a buffer overflow, and can thus be computed as $V^B(t) = B + V(t)$.

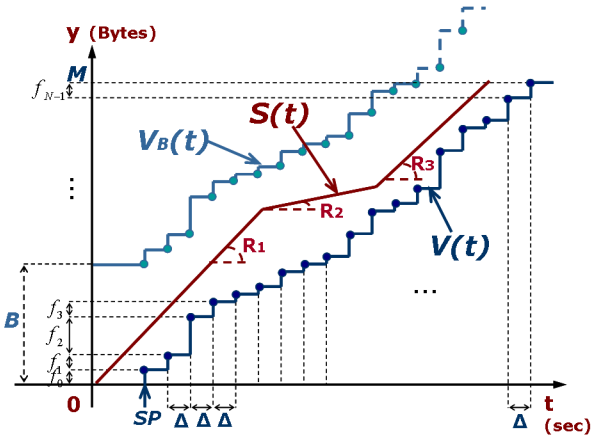


Figure II.2.: Streaming model under real-time constraints: graphical model.

Table II.1.: Symbols used in the Streaming Model.

| Symbol | Meaning |
|---|---|
| $\Delta$ | The time interval between two consecutive frames. |
| $N$ | The number of frames in the multimedia file. |
| $f_i,\ 0 \leq i < N$ | The size of the $i^{th}$ frame in the multimedia file. |
| $M = \sum_{i=0}^{N-1} f_i$ | The size of the multimedia file. |
| $B \geq \max_{0 \leq i < N} f_i$ | The size of the client buffer. |
| $S(t)$ | The scheduling function. |
| $R_i$ | The $i^{th}$ transmission rate. |
| $SP$ | The Start Point: when playback begins. |
| $V(t)$ | The playback curve. |
| $V^B(t)$ | The upper bound constraint to the scheduling function. |

The *scheduling function* $S(t)$ defines the number of bytes sent by the proxy to the client by time $t$, while the corresponding arrival function $A(t)$ defines the number of bytes received by the client by time $t$. Clearly, the relation $S(t) \geq A(t)$ always holds, the difference between $S(t)$ and $A(t)$ depending on the network delay and jitter. However, it is generally assumed that $S(t) = A(t)$ and the difference between the two curves is heuristically taken into account by slightly enlarging the size of the client buffer so as it can *absorb* both the jitter and the transfer delay. Therefore, hereafter we will assume $S(t) = A(t)$. Based on this assumption, and recalling the definition of both $V(t)$ and $V^B(t)$, it follows that a scheduling function is any non-decreasing function confined between $V(t)$ and $V^B(t)$, i.e., the following equation holds:

$$V(t) \leq S(t) \leq V^B(t). \tag{II.2}$$

Finally, it is worth noting that the playback process extracts one single frame at a time from the client buffer at each playback time. Therefore, the level of the client buffer, at the time instant $t$, is the difference between the total number of bytes arrived at the client by time $t$, and the total number of bytes already played back, i.e., $S(t) - V(t)$.

The scheduling algorithms define different scheduling curves depending on the particular goal targeted. When traffic smoothing is the target, the scheduling curve is a polyline with line segments of different slopes corresponding to different transmission rates ($R_i$ is the $i^{th}$ transmission rate). In this chapter we will use polylines that alternate line segments with zero slope (OFF periods) and line segments with non-zero slope (ON periods).

## 5.2. Start Point Computation

In this section we will show how the *RT_PS* protocol derives the start point, i.e., the initial playback delay. As a first step, it is worth focusing on constant-rate ON/OFF schedules, i.e. ON/OFF schedules where data transfers during the ON periods occur at a constant rate $R$. A deep analysis on constant-rate ON/OFF schedules applied to VBR traffic can be found in [Zha96] and [ZH97]. These pieces of work prove that, given the constrained region delimited by the functions

$V(t)$, $V^B(t)$, $t = 0$, $y = 0$ and $y = M$ (see Fig. II.2), it is possible to find an ON/OFF schedule only in case the transmission rate is greater than a minimum value, hereafter referred to as $\bar{R}$. For some combination of $V(t)$, $V^B(t)$, and $M$, $\bar{R}$ could be $0$, meaning that the schedule is always feasible. For any $R \geq \bar{R}$, many ON/OFF schedules can be conceived, which differ in the start point value, and the number, position and duration of the OFF periods. However, a *minimum* ON/OFF schedule exists among these, which is defined as *R-envelope* and denoted as $E_R(t)$. It was formally defined in [Zha96] and is, intuitively, the ON/OFF schedule closest to the playback curve $V(t)$. Therefore, for any possible ON/OFF schedule $S_R(t)$ the following equation holds:

$$S_R(t) \geq E_R(t); \tag{II.3}$$

$\forall t,\ 0 \leq t < SP + N\Delta,$ , where $N$ is the total number of frames composing the file, and $\Delta$ is the time the playback process takes to play a single frame. For the sake of semplicity, hereafter we will only refer to the sequence $E_R[k]$ [HHHK02] instead of the continuous function $E_R(t)$. $E_R[k]$ is composed by the samples picked up from $E_R(t)$ at regular time intervals. It can easily be obtained starting from the definition of its last sample $E_R[N-1]$ which corresponds to the end of the playout curve ($M$) at the last playback time. Formally, $E_R[N-1] = V(t^{last}) = M$, where $t^{last} = SP + (N-1)\Delta$. Drawing from there a descending *R-sloped* line segment that ends at the previous playback time (i.e., $\Delta s$ before, when $t = t^{prev} = SP + (N-2)\Delta$), the new sample $E_R[N-2]$ corresponds to the end-point of the segment, in case it is higher than the value of the playout curve $V(t^{prev})$, or to the corresponding value of the playout curve $V(t^{prev})$ otherwise. Subsequent samples can be obtained, again, starting from the new sample $E_R[N-2]$ and drawing a new *R-sloped* line segment till the previous playback time, and so on. By repeating this procedure until the time $t = 0$ is reached, the entire sequence $E_R[k]$ is obtained. More

formally, $E_R[k]$ can be expressed as follows:

$$E_R[k] = E_R(SP + k\Delta) =$$

$$\begin{cases} V(SP+(N-1)\Delta) & if\ k = N-1 \\ max\{E_R[k+1] - R_\Delta,\ V(SP+k\Delta)\} & if\ 0 \leq k < N-1 \\ 0 & if\ k < 0 \end{cases} \quad \text{(II.4)}$$

where $0 \leq k < N$ and $R_\Delta$ is the total number of bytes that can be transmitted during the time interval $\Delta$ assuming a transmission rate $R$.

The minimum rate $\bar{R}$ that allows an ON/OFF schedule to be performed is exactly the minimum rate for which an $E_R(t)$ curve exists inside the region defined by $V(t)$, $B$, and $M$. Specifically, the following equation holds:

$$\bar{R} = min\{R \mid \exists E_R(t)\}. \quad \text{(II.5)}$$

At the beginning of the streaming session, the *RT_PS* proxy-side component calculates the minimum rate $\bar{R}$ for the audio sequence to be transmitted, and contrasts it with the bandwidth $R^0$ available on the wireless link. The streaming process can actually start only if $R^0 \geq \bar{R}$.

After this preliminary check, the *RT_PS* proxy component can decide the playback *start point*. Assuming that the proxy starts to transmit the frames at time $0$, the start point corresponds to the initial delay. Obviously, a long initial delay grants more time for *pre-buffering* at the client and prevents the playback process from starving due to buffer underflow. However, several reasons suggest keeping the initial delay short. First, the user is not willing to wait for long time before starting to listen to the selected audio file. Moreover, as is shown in [Zha96], an increase in the initial delay causes an increase in the total idle period (i.e., the sum of all the OFF periods) while the total ON period remains unmodified. In our solution this increases the energy consumption due to the energy amount consumed by the WNIC while in the sleep mode.

The start point is calculated as follows. Given the initial transmission rate $R^0$, the minimum possible initial delay $d_{R^0}$ is the distance, measured on the $X$-*axis* $(t)$,

between $R^0$-*envelope* $E_{R^0}(t)$ and $V(t)$. To avoid underflowing the client buffer due to possible variations in the available throughput, the initial delay is calculated with reference to $R^0/2$-*envelope* instead of $R^0$-*envelope*, i.e., $d_{R^0/2}$ is used instead of $d_{R^0}$. The rationale behind this heuristic comes from the following property [Zha96]. Given two transmission rates $R_1$ and $R_2$, such that $0 \leq R_1 \leq R_2$, the following relations hold:

$$E_{R_1}(t) \geq E_{R_2}(t),\ d_{R_1} \geq d_{R_2}; \tag{II.6}$$

$\forall\ t,\ 0 \leq t < SP + N\Delta$. Using $d_{R^0}$ would lead to buffer underflow even for small reductions in the available throughput. On the other hand, using $d_{R^0/2}$ a buffer underflow does not occur unless the available throughput falls below $R^0/2$.

Finally, for very low values of $R^0$, the initial delay $d_{R^0/2}$ may be too long, beyond the maximum delay the user is willing to tolerate. In this case the initial delay is taken as the maximum initial delay specified by the user.

## 5.3. Sleep Time Computation

The next issue to be addressed is how to compute the duration of an OFF period during which the WNIC can be switched to the sleep mode. An OFF period starts when either the client buffer fills up, or the available throughput becomes too low (half the initial value).

The procedure described above to calculate the initial delay is used to compute the durations of the OFF periods too. Let us assume that the server has just scheduled and sent to the client $S(t^*)$ bytes. Let us also assume that, according to the last estimate provided by the client, the current transmission rate (available throughput) on the wireless link is $R^*$. Then, the maximum allowed sleep time corresponds to the distance, measured on the $y = S(t^*)$ horizontal axis, between $S(t^*)$ and $R^*$-*envelope* $E_{R^*}(t)$. Intuitively, assuming that the throughput $R^*$ will also be available in the near future, the mobile host can sleep until the scheduling function intersects $E_{R^*}(t)$. As above, to reduce the risk of playback starvation, $R^*/2$-*envelope* $E_{R^*/2}(t)$ is used instead of $R^*$-*envelope* $E_{R^*}(t)$. Therefore, a buffer underflow will not occur unless the available throughput falls down $R^*/2$.

Obviously, an OFF period actually starts only if the calculated OFF duration is greater than the time needed to the WNIC to switch from active to sleeping and back from sleeping to active. When this condition does not hold the client buffer level is deemed to be a *low water* level because the scheduling function is very close to the playback curve and a transmission interruption is too dangerous. In this case, the transmission must continue even if the available throughput on the wireless channel is low. Otherwise, the client buffer level is considered safe and an OFF period can start.

## 6. Available Throughput Estimate

In this section we will show how the *RT_PS* protocol estimates the throughput available on the wireless link. The transmission rate used by the *RT_PS* protocol to derive the ON/OFF schedule corresponds to the estimated available throughput.

It is yielded with a very simple and fast technique that guarantees good accuracy for our purposes nevertheless. In principle it works as follows. The *RT_PS* proxy sends a train of *back-to-back* packets to the *RT_PS* client. The packet train is preceded by a special packet informing the client that a new packet train is starting. Upon receipt of the special packet the client records the time. Moreover, it starts counting the number of bytes contained in the subsequent packet train. The available throughput experienced by the client is thus estimated as the ratio between the total number of bytes contained in the received packet train and the total time needed to receive those bytes (time interval between receipt of the special packet and receipt of the last packet in the train).

For an accurate estimate two key factors must be taken into account: the *packet train length* and the *size of each single packet*. In our experiments we found that the best accuracy is achieved when the packet size is $1472$ bytes. This value corresponds to the maximum UDP-payload size that does not undergo the MAC layer fragmentation (when the packets transmitted during the streaming session are shorter than $1472$ bytes, the overhead times introduced for each single transmission at the MAC layer -according to the 802.11b standard- have a greater impact on the total transmission time and the resulting throughput calculated). We also

found that, to achieve an accurate estimate the packet train should include at least 50 1472-*byte* packets. Under these conditions, we found that the maximum throughput experienced by the client in a Wi-Fi WLAN without any other competing traffic is approximately 6.65 Mbps when the WNIC data rate is 11 Mbps, 4.04 Mbps when the data rate is 5.5 Mbps, and 1.69 Mbps when the data rate is 2 Mbps.

However, in our implementation, estimates work out from the *RT_PS* traffic during the ON periods. Hence, a packet is an *RT_PS* message that includes an RTP packet. The RTP packet is further composed by couples *<ADU descriptor, ADU frame>* that cannot be fragmented for the sake of loss tolerance [Fin04]. This may lead to a large variability in the packet sizes and to degrading the accuracy in the estimates. To overcome this problem, the *RT_PS* proxy sends an integer value of couples *<ADU descriptor, ADU frame>* in a single RTP packet so as to keep the *RT_PS* packet sizes as close as possible to the ideal value (i.e., 1472 bytes). Moreover, the train length is increased from 50 to 60 packets. This is generally possible in our scenario except when the client buffer is small. In that case the accuracy in estimates slightly decreases. Fig. II.3 shows the throughput estimated on a

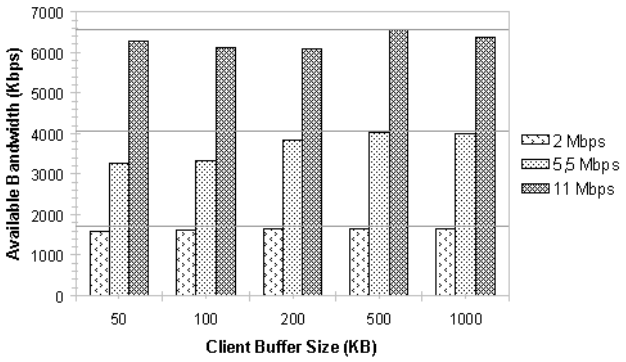


Figure II.3.: Available bandwidth estimates for different client buffer size and WNIC data rate.

WLAN without competing traffic, for different data rates of the WNIC and different sizes of the client buffer. These results have been obtained by using a prototype implementation of the proposed architecture (see Section 7 for details about the

experimental testbed and the measurement methodology). Specifically, the above algorithm was used for estimating the throughput experienced by the client. The results in Fig. II.3 show that, in absence of interfering traffic, a streaming session is able to exploit up to $6.54$ Mbps (instead of $6.65$ Mbps) with $11$ Mbps data rate, up to $4.01$ Mbps (instead of $4.04$ Mbps) with $5.5$ Mbps data rate, and up to $1.66$ Mbps (instead of $1.69$ Mbps) with $2$ Mbps data rate. These results show that our estimate methodology, based on real *RT_PS* traffic is accurate.

The client buffer size is another factor that affects the accuracy in the throughput estimates. When the client buffer is large, the *RT_PS* schedule produces a small number of very long OFF periods. This results in a very bursty traffic where transmissions are concentrated in short periods separated by long interruptions. As the client buffer size decreases the number of the OFF periods increases, whereas their lengths decrease. Therefore, more control packets are needed [ACG+05b], which are shorter than the packets carrying audio data. Clearly, this results in the available throughput be slightly underestimated, as highlighted in Fig. II.3. Based on the above experimental results, we can conclude that, for sufficiently large buffers, our algorithm for bandwidth estimate is accurate. Please note that, since each estimate is derived from a long train of packets, short-term variations in the available throughput are filtered out.

We also investigated the effect of some competing traffic on the accuracy of the estimate algorithm. When adding background traffic to the streaming flow the estimates of the available bandwidth decrease accordingly. Fig. II.4 focuses on a WNIC data rate of $2$ Mbps and shows the estimated throughput for increasing rates of the background traffic and different client buffer sizes. As expected, the available throughput decreases as the background traffic grows up. When the background traffic is so high to saturate the wireless bandwidth both the background and the streaming traffic are expected to achieve half the maximum available bandwidth, i.e. $845$ Kbps (as is stated above, the maximum available bandwidth is $1.69$ Mbps when the data rate of the WNIC is $2$ Mbps). Fig. II.4 shows that the throughput estimated is very close to the expected results. Again, the best accuracy is achieved with the largest client buffer size.

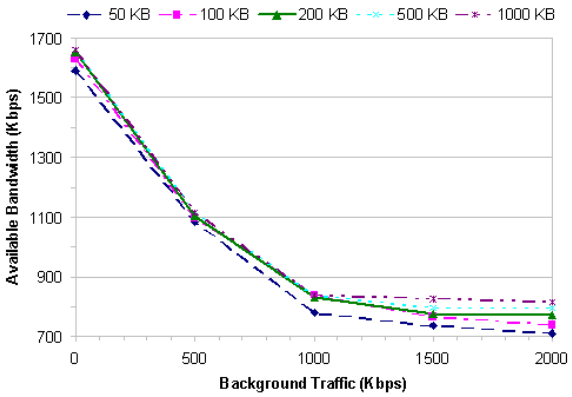


Figure II.4.: Available bandwidth estimates for different client buffer size and background traffic.

## 7. Experimental Evaluation

We implemented the *RT_PS* protocol in a prototype version of our architecture and conducted an extensive experimental analysis to evaluate its performance. In our analysis we first considered a scenario with a single streaming session from the proxy to the mobile client. Then, we also considered the more general case with multiple simultaneous streaming sessions.

We evaluated the energy efficiency of our protocol by measuring the power saving index $I\_ps$ which is the ratio between the energy consumed during the streaming session by the WNIC of the client device when the *RT_PS* protocol is used, and the energy consumed without the *RT_PS* protocol [ACGP03b], [ACGP03a]. More formally, $I\_ps$ is defined as follows:

$$I\_ps = \frac{E_{RT\_PS}}{E_{\overline{RT\_PS}}}; \tag{II.7}$$

where:

$$E_{RT\_PS} = (T_{ON} \cdot W_{ON}) + (T_{SLEEP} \cdot W_{SLEEP}), \tag{II.8}$$

and

$$E_{\overline{RT\_PS}} = W_{ON} \cdot (T_{SLEEP} + T_{ON}). \quad \text{(II.9)}$$

In Equations II.8 and II.9 $T_{ON}$ is the sum of all the ON periods, while $T_{SLEEP}$ is the sum of all the OFF periods. The power consumption of the WNIC in the active mode was approximated by a constant value, irrespective of the specific state (rx, tx, idle) the WNIC was operating in. Specifically, we considered $W_{ON} = 750mW$ and $W_{SLEEP} = 50mW$ [KB02].

Finally, it should be pointed out that each experiment was replicated three times, and the results reported below are the average values over the three replicas.

## 7.1. Experiences with a single streaming session

In the first set of experiments we considered a single streaming session from the proxy to the client, and investigated the performance of the *RT_PS* protocol for different data rates of the WNIC (we manually set the WNIC to work at a constant data rate of either $2$ Mbps, $5.5$ Mbps or $11$ Mbps during each single experiment) and different sizes of the client buffer (from $1000$ KB down to $50$ KB). We also analysed the impact on the performance of some background traffic.

When there is no background traffic the throughput achieved by the streaming session is maximum, and the proxy stops transmitting data to the client only when the client buffer is full. As expected, the *RT_PS* protocol performs better for greater data rates. This is because when the data rate is greater the data transfer is faster, and the WNIC is active for shorter time. We found that, with a WNIC data rate of $11$ Mbps, $I\_ps$ is almost constant (i.e., it exhibits only a slight dependence on the client buffer size), and ranges from $8.50\%$ ($10$ MB) to $8.63\%$ ($50$ KB). Hence, the *RT_PS* protocol allows up to $91.50\%$ saving of the energy consumed when no energy management policy is used. When the WNIC data rate decreases, the $I\_ps$ index increases, i.e., the energy saving decreases. With a WNIC data rate of $5.5$ Mbps, the $I\_ps$ index ranges from $9.66\%$ ($10$ MB) to $10.33\%$ ($50$ KB), and with a data rate of $2$ Mbps it ranges from $13.90\%$ ($10$ MB) to $14.23\%$ ($50$ KB). In the worst case the energy saving is greater than $85\%$. The above results show that the $I\_ps$ index increases as the client buffer size decreases. These variations can be

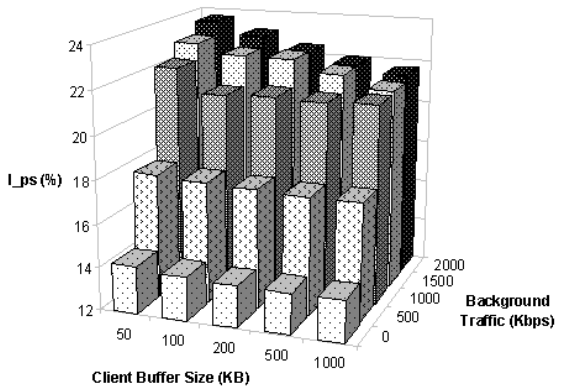


Figure II.5.: $I\_ps$ experienced by a streaming flow competing with CBR background traffic. Min burstness: 1 packet per single burst.

explained as follows. As anticipated in Section 6, when the buffer size decreases the following side effects occur: i) the amount of control traffic increases, ii) the accuracy of the estimator of the available bandwidth decreases, and iii) the schedule is characterized by a *greater* number of *shorter* ON periods. Due to i) the WNIC consumes more energy because is active for longer time to support the increased control traffic. Moreover, since for small buffer sizes the estimator tends to underestimate the available bandwidth, the WNIC remains active for much more time in order to prevent buffer underflows. Finally, a greater number of short ON periods in the transmission schedule contributes to increase the energy consumed in frequently switching the WNIC on and to sleep mode.

To investigate the impact of the interfering traffic on the protocol performance we introduced an additional mobile host sending CBR (Constant Bit Rate) traffic, i.e., a periodic flow of UDP packets, to the proxy. We tested our system in the worst case, i.e., with a WNIC data rate of 2 Mbps. Fig. II.5 summarizes the results obtained from the experiments with the background CBR traffic. It clearly emerges that $I\_ps$ increases when the amount of background traffic increases, thus leading to a higher energy consumption. This is because the streaming session experiences a lower throughput and the WNIC must be active for longer time since the client

Table II.2.: $I\_ps$ vs. Client Buffer Size. Worst Case: $2$ Mbps Data Rate, $2$ Mbps Background Traffic.

| **Client Buffer Size (Kbytes)** | **Average Transmission Rate (Mbps)** | $I\_ps$ **(%)** |
|---|---|---|
| 1000 | 0.815 | 21.62 |
| 500 | 0.795 | 21.99 |
| 200 | 0.773 | 22.49 |
| 100 | 0.738 | 23.06 |
| 50 | 0.711 | 23.65 |

buffer fills up slowly and the risk of playback starvation is higher. When the rate of the background flow increases beyond $845$ Kbps the wireless bandwidth is almost equally shared between the streaming flow and the CBR flow, i.e., the throughput experienced by the streaming session is approximately constant ($845$ Kbps is half of the maximal bandwidth, $1.69$ Mbps, experienced with $2$ Mbps data rate). Therefore, for a given buffer size, $I\_ps$ tends to flatten. Table II.2 shows the performance of the *RT_PS* protocol in the worst conditions, i.e., when the rate of the background traffic is $2$ Mbps and, hence, the interfering mobile host has always a packet ready for transmission. $I\_ps$ is, at most, equal to $23.65\%$ which corresponds to an energy saving of $76.35\%$.

To conclude our analysis on the impact of the background traffic, we now show the evolution over time of the following parameters: client buffer level, ON and OFF durations, and estimated bandwidth. Fig. II.6 shows the system behavior when there is no background traffic, while Fig. II.7 shows the same behavior when the background traffic is maximum. Fig. II.6 shows that the estimated bandwidth (the dots inside the ON periods) is high because there is no interfering traffic, and the buffer level grows up until the buffer becomes full. During the OFF periods the playback process consumes data in the client buffer at a constant rate and the buffer level decreases linearly. In Fig. II.7 the bandwidth estimated is lower because of the background traffic. Therefore, the buffer level during the ON periods

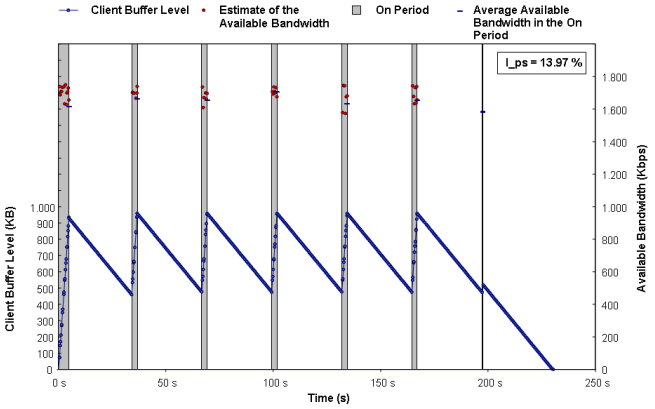


Figure II.6.: Variations in the client buffer level during the streaming session. The client buffer size is 1000 KB. No background traffic is present.

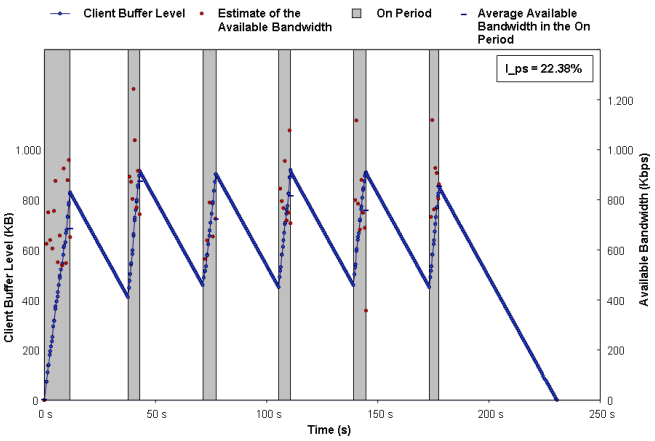


Figure II.7.: Variations in the client buffer level during the streaming session. The client buffer size is 1000 KB. Maximal background traffic is present.

increases at a slower rate than in Fig. II.6. Moreover, the ON periods are longer compared with those in Fig. II.6. Finally, there is a higher fluctuation in the es-

Table II.3.: $I\_ps$ vs. Burst Sizes in the Background Traffic. $1$ Mbps Background Traffic, $2$ Mbps Data Rate

| **Buffer** | **Burst Size (pkts)** | | | | |
|---|---|---|---|---|---|
| **Size(KB)** | **1** | **10** | **50** | **100** | **200** |
| 1000 | 21.21 | 20.27 | 18.76 | 18.27 | 16.70 |
| 500 | 21.15 | 20.65 | 18.82 | 19.20 | 16.51 |
| 200 | 21.23 | 20.72 | 18.49 | 17.69 | 17.47 |
| 100 | 21.13 | 20.44 | 16.76 | 14.76 | 18.05 |
| 50 | 22.22 | 20.65 | 18.20 | 18.65 | 17.71 |

timates of the available bandwidth. Nevertheless, the available bandwidth is on average about $845$ Kbps, i.e., half the total available bandwidth. This proves that the bandwidth is equally shared by the two competing traffic flows.

Finally, we investigated the effects on $I\_ps$ of the *burstiness* in the background traffic. To this end we conducted a set of experiments where the CBR traffic generator delivered a *burst* of packets at a time instead of a single packet at a time. We used burst sizes of $10$, $50$, $100$, $200$ packets in our experiments. Since the rate of the background traffic is kept constant, an increase in the burst size results in a longer inter-time between two consecutive bursts.

Table II.3 shows the $I\_ps$ values for different burst and client buffer sizes. These results were obtained with a WNIC data rate of $2$ Mbps and background traffic of $1$ Mbps. They show that as the burst size increases the energy efficiency increases accordingly. This is because the streaming session takes advantage of the non-transmission periods in the background traffic. When the *RT_PS* proxy component accesses the wireless channel, it may be either busy or free depending on whether the generator of the background traffic is producing a burst or not. If the generator is silent the proxy starts transmitting immediately. On the other hand, if a burst is flowing up the *RT_PS* protocol estimates a low available bandwidth and the

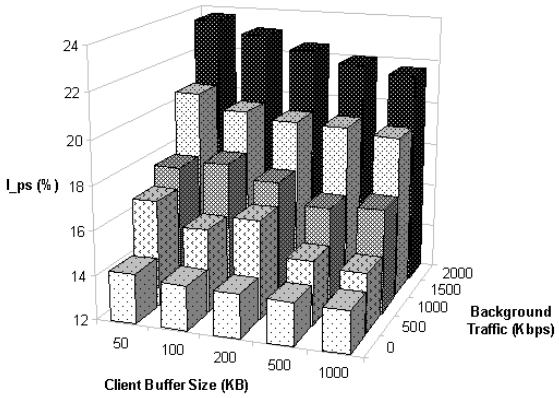


Figure II.8.: $I\_ps$ experienced by a streaming flow competing with a bursty CBR traffic. Max burstness: $200$ packets per single burst.

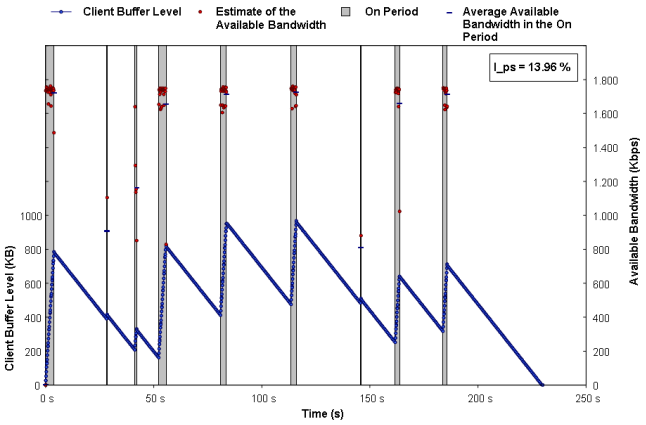


Figure II.9.: Variations in the client buffer level during the streaming session. The client buffer size is $1000$ KB. The background traffic is $500$ Kbps with $200$ packets sent per single burst.

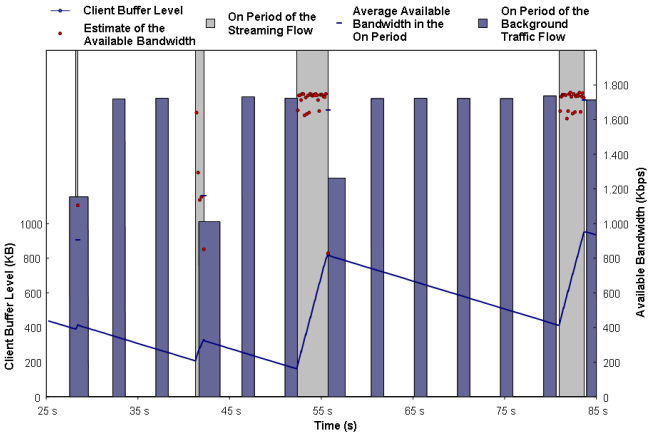


Figure II.10.: Variations in the client buffer level during the streaming session from 25 s to 85 s. The client buffer size is 1000 KB. The background traffic is 500 Kbps with 200 packets sent per single burst.

proxy defers the transmission. Finally, if a new burst starts in the middle of a transmission causing the estimated available bandwidth to go down, the proxy stops transmitting.

Intuitively, for a given background traffic rate, an increase in the burst size leads to longer time intervals between consecutive bursts. During such intervals the proxy can exploit the maximum bandwidth to fill up the client buffer, and hence the performance approaches the case without background traffic. The best condition occurs when the proxy finds the channel free for enough time to fill up the client buffer. The streaming flow and the background traffic may become somewhat synchronized, i.e., the ON periods of the streaming flow fit into the silent periods of the background traffic and vice-versa. In this case, both the streaming and the background traffic flows can thus exploit the maximum available bandwidth.

Fig. II.8 summarizes the results obtained from the set of experiments with the bursty background traffic. By contrasting Fig. II.8 with Fig. II.5, it clearly appears that higher energy efficiency is achieved when the burstiness in the background traffic increases.

Fig. II.9 shows the client buffer level evolution during an entire streaming session in presence of background traffic of $500$ Kbps. The background traffic generator generates $200$ packets per burst. The client buffer size is $1000$ KB. It appears that the *RT_PS* protocol is able to adapt to the background traffic behavior as the ON periods of the streaming session correspond almost always to the idle periods in the background traffic (the available bandwidth estimated during the ON periods is very high). To better highlight this behavior a portion of Fig. II.9 is magnified in Fig. II.10 where both the ON periods in the streaming session and the idle phases in the background traffic are shown. The first ON period in the streaming flow starts when a burst is in progress. The available bandwidth estimated by the *RT_PS* protocol is thus low and the mobile client enters the sleep mode. The second ON period begins during an idle phase in the background traffic. However, the traffic generator starts sending a new burst soon later. As soon as the estimated bandwidth falls below a threshold the *RT_PS* protocol decides to defer the transmission. Finally, during the third and fourth periods no significant overlap is experienced, and data flows at the maximum speed. In conclusion, the *RT_PS* protocol is able to adapt to the background traffic behavior. The throughput experienced during the ON periods is thus very high, as if there was no competing traffic on the channel, and the energy efficiency is very good.

## 7.2. Experiences with two simultaneous streaming sessions

In this section we will analyse the performance of the *RT_PS* protocol when there are more simultaneous streaming sessions. Specifically, in our experiments we will consider two concurrent streaming sessions with clients running on two different mobile hosts. In some experiments we will also consider a third mobile host generating interfering traffic.

Fig. II.11 shows the $I\_ps$ index achieved by both clients when no background traffic is injected in the network. In this set of experiments the two streaming sessions start at the same time. The data rate of the WNICs is $2$ Mbps. As is highlighted in the figure, there is no significant difference in the performance achieved by the two sessions. At a smaller scale, results show a slightly favorable trend towards bigger buffer sizes, as expected. Anyway, we can conclude that the energy saving

achieved by both clients is always high. It is worth pointing out that, when there is no background traffic, no streaming session undergoes contention over the wireless LAN, since both the streaming traffic flows flow from the proxy to the client. However, depending on the particular streams and on their starting points, they may or may not achieve the $I\_ps$ value experienced in the same conditions by a single streaming session alone. Actually, when the ON periods of both streams overlap, the proxy has to schedule packets for both of them. By assuming a simple round-robin policy, both streams will experience half the maximum bandwidth. As anticipated in the previous section, this leads to a higher energy consumption. On the other hand, both streams achieve the maximum energy saving when the ON periods do not overlap. To conclude our analysis, we considered the impact

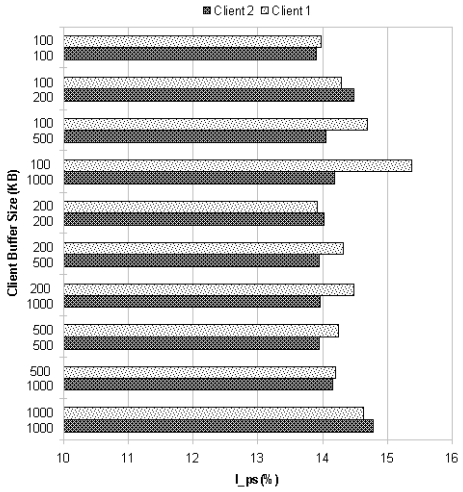


Figure II.11.: Two competing streaming flows with no background traffic: $I\_ps$ experienced when combining different client buffer sizes for the two flows.

of some background traffic also in this second scenario. We considered both UDP and TCP interfering traffic flows (see Fig. II.12). As expected, when introducing UDP traffic the $I\_ps$ index of both the streaming flows increases (i.e., the energy efficiency decreases). The worst case is experienced when the background traffic reaches half the total bandwidth of the wireless channel. However, there are not

significant differences in the $I\_ps$ values achieved by the two streaming flows. When the interfering traffic is TCP, the $I\_ps$ value experienced by the two streaming flows slightly worsens. The additional bandwidth required by the TCP ACKs can be accounted for this behavior. Again, there is no significant difference in the $I\_ps$ values achieved by the two streaming flows.

The above analysis was aimed at characterizing the energy efficiency of our system. However, in a multimedia streaming application the frame loss and initial playback delay are two important performance figures that need to be taken into consideration as well. In all our experiments the initial playback delay was always less than $2s$. Moreover, the fraction of frames discarded by the proxy (because expected to arrive late for playback) or lost during the transmission was always below $5\%$ of the total number of frames. This percentage is considered largely acceptable since, in real-time audio streaming, up to $20\%$ loss can be tolerated if loss concealment techniques are implemented at the receiver.

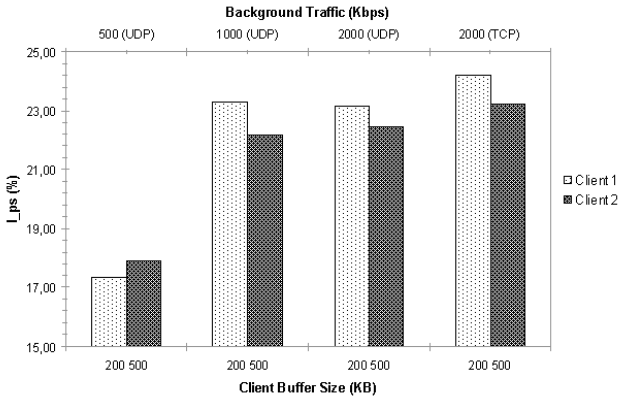


Figure II.12.: $I\_ps$ experienced in two competing streaming flows when adding background traffic of different types.

# 8. Conclusions

In this chapter we have considered a mobile wireless scenario for multimedia streaming applications, and proposed an architecture for providing energy-efficient streaming services to mobile clients. Specifically, we have focused on MP3 audio streaming in a Wi-Fi environment. Our architecture is based on the server-proxy-client model, where the proxy functionality is implemented at the Access Point (or at a fixed host on the same LAN of the Access Point). The architecture relies on the *RT_PS* (*Real Time and Power Saving*) protocol between the proxy and the mobile client. It implements an energy management policy whose main goal is to minimize the energy consumed by the wireless network interface at the mobile host, while guaranteeing the real-time constraints of the streaming application. The protocol uses an adaptive ON/OFF schedule for the transmission of audio frames to the mobile client, and switches the wireless network interface to the low-power sleep mode during the OFF periods. The *RT_PS* scheduling policy relies on the availability of a buffer at the client where frames can be stored before their playback times, and is mainly based on the following heuristics. When the available bandwidth on the wireless link is high the proxy transmits data until the client buffer becomes full. When the available bandwidth is low -due to competing traffic on the wireless network- the proxy defers the data transmission to better times. However, if the client buffer level becomes too low the proxy transmits data irrespective of how much bandwidth is available. To evaluate the performance of our solution in a real environment we implemented our *RT_PS* protocol in a real prototype system based on the Wi-Fi technology. The results obtained by means of an extensive experimental analysis have shown that our solution can save up to $91.50\%$ of the energy consumed by the Wi-Fi interface at the mobile host when no energy management policy is used. The energy efficiency of the protocol proposed depends on the data rate of the wireless interface (it increases as the data rate increases) and the client buffer size (it increases as the buffer size increases). However, even with small buffer sizes (e.g., $50$ KB) the energy saving is very high. We also investigated the impact of some interfering traffic on the system performance. Even though the competing traffic reduces the energy efficiency, the measured energy saving is never less than $75\%$. Interestingly, we also

found that the energy efficiency increases when the burstiness in the interfering traffic is higher.

# III. MANETs for Real: Results from a Small-scale Testbed

## 1. Introduction

In the last ten years there has been a large volume of research activities devoted to mobile ad hoc networking (see [CCL03], and [BCG04]). However, almost all research areas (from enabling technologies to applications) still harbor many open issues. This is characteristically exemplified by research activities performed on routing protocols. Most work on routing protocols is being performed in the framework of the IETF MANET working group, where four routing protocols are currently under active development. These include two reactive routing protocols, AODV and DSR, and two proactive routing protocols, OLSR and TBRPF [Roy04]. There has been good progress in studying the protocols' behavior (almost exclusively by simulation), as can be seen in the large conference literature in this area, but the absence of performance data in non-trivial network configurations continues to be a major problem. There are two main approaches in system performance evaluation: the first is based on a representation of the system behavior via modeling [Lav83] [KM88], the second uses measurements. Measurement techniques are applied to real systems, and thus they can be applied only when a real system, or a prototype of it, is available. For this reason, most results on the behavior of MANET protocols have been obtained by defining a system model, and solving the model using analytical and/or simulative techniques. Analytical models are often not detailed enough for the ad hoc networks evaluation. On the other hand, simulation modeling is a more standardized, mature, and flexible tool for modeling various protocol and network scenarios. It allows (by running the simulation

model) collection and analyses that fully characterize the protocol performance in most cases. A very large number of simulation models have been developed to study ad hoc network architectures and protocols under many network scenarios, number of nodes, mobility rates, etc., see [BB04]. Simulation studies have been extensively applied for instance to compare and contrast large number of routing protocols developed for MANETs, see e.g., [JLHM99] [DCY00] [DPR00] [BMJ+98].

As pointed out in [GLNT], simulations have not been conclusive for selecting MANET protocols among the several available solutions. Furthermore, simulation models often introduce simplifications and assumptions that mask (in simulation experiments) important characteristics of the real protocols behavior [ABCG04] [LNT02], see for example, the so-called *communication gray zones* problem [LNT02]. This problem was revealed by a group of researchers at the Uppsala University, while measuring the performance of their own implementation of the AODV routing protocol in an IEEE $802.11b$ ad hoc network. Observing an unexpected large amount of packets' losses, mainly during route changes, it was found that increase in packet loss occurred in some specific geographic areas called *communication gray zones*[1]. It is important to point out that the communication-gray-zone problem cannot be revealed by commonly used simulation tools (e.g., NS-2, Glomosim) as in their $802.11$ models both unicast and broadcast transmissions are performed at 2 Mbps, and hence have the same transmission range. To avoid these modeling approximations, simulations have to be complemented by experiments on real prototypes. In addition, the availability of prototypes will also make possible to start creating communities of MANET users that, by experimenting with this technology, will provide feedback on its usefulness and stimulate the development of applications tailored for the ad hoc environment. User interaction can finally result in teh identification of possible ad-hoc-network killer applications making MANETs a success beyond the academic world. Currently, only a few measurement studies on real ad hoc testbeds can be found in the literature, see e.g., [BMJ00] [WSi]. The Uppsala University APE testbed [WSi] is one of the

[1] This phenomenon is due to the different transmission ranges between unicast (data) and broadcast frames (i.e., routing information) in $802.11$ networks. A station inside a gray zone is considered reachable by a neighboring station using the routing information, while actual data communication between the stations is not possible.

largest, having run tests with more than thirty nodes. Results from this testbed are very important [GLNT] and point out that more research in this direction is required to consolidate the ad hoc networking research field. In the framework of our research activities to investigate potentialities and limits of the ad hoc networking paradigm we set up a MANET prototype on which we performed a set of measurement activities. In a previous phase [ABCG04] we focused on single layers of an ad hoc network (mainly studying 802.11 performance in ad hoc networks). This chapter presents measurement results obtained by implementing a full MANET protocol stack. From this point of view our work extends the analysis presented in [GLNT] where the focus was on layers 1-3 (with a special attention to routing protocols). The chapter is organized as follows. In Section 2 we present the architecture and protocols used in our experiments. Section 3 describes the scenario in which experiments were performed. Preliminary experimental results (warm-up phase) are described in Section 4. Section 5 presents an in depth analysis of the more interesting phenomena identified in the warm-up phase. Lessons learned from these experiments are discussed in Section 6.

## 2. Testbed Reference Architecture

Measurement studies are often focused on a single aspect of the MANET world, mainly routing protocols. The novel aspect of our study is to investigate a full ad hoc network architecture, from the Wi-Fi ad hoc network up to the peer-to-peer middleware platform (*Pastry*) on top of which a simple testing application (distributed messaging) is running. For our experimentation we used the reference architecture shown in Figure III.1. More precisely, we integrated solutions for the main building blocks that makes our testbed representing a realistic MANET: (i) Enabling Technologies, (ii) Networking, and (iii) Middleware and Applications.

**Enabling Technologies**
The success of a network technology is connected to the development of networking products at a competitive price. A major factor in achieving this goal is the availability of appropriate networking standards. Two main standards are emerging for ad hoc wireless networks: the IEEE 802.11 standard for WLANs [WSf],

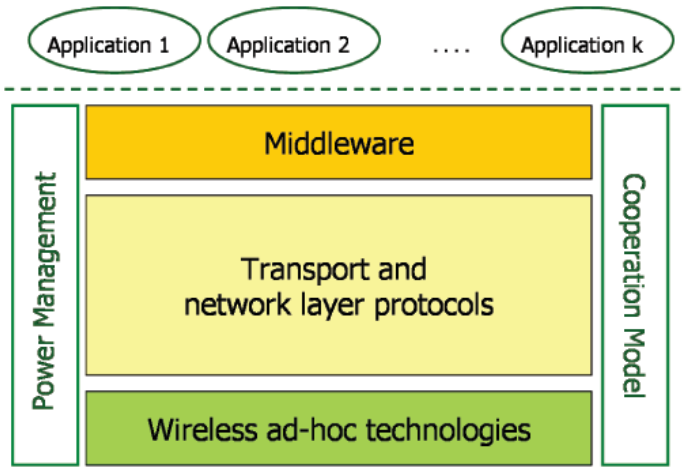


Figure III.1.: Reference Architecture

and the Bluetooth specifications [WSn] for short-range wireless communications [Bis01]. [ZD04] surveys the off-the-shelf technologies for constructing ad hoc networks. Due to its extreme simplicity, and its widespread usage, the IEEE $802.11b$ is the technology we selected for constructing our multi-hop ad hoc network. [ACG04a] and [ABCG04] present an in depth analysis of $802.11$-based ad hoc networks, including measurement-based performance evaluation.

**Networking**

The aim of the networking protocols is to use the one-hop transmission services provided by the enabling technologies to construct end-to-end (reliable) delivery services, from a sender to one (or more) receiver(s). Routing and forwarding algorithms must thus be provided to route the information through the MANET. In addition, the low reliability of communications (due to wireless communications, users' mobility, etc.), and the possibility of network congestion may require Transport Layer mechanisms to guarantee a reliable end-to-end information delivery. To implement these functionalities in a MANET, beyond the legacy Internet protocols (IP, TCP and UDP), routing and forwarding functionalities tailored on the characteristics of this environment are required. Today, among the four routing protocols currently under active development inside the IETF MANET working

group [Roy04], AODV and OLSR are the most mature routing protocols from the implementation standpoint; for the other MANET protocols either updated implementations are not available (DSR) or there is no freely available implementation (TBRPF). For this reason, we integrated in our test-bed the implementation of one reactive (AODV) and one proactive (OLSR) routing protocol. Furthermore, some DSR experimentations are carried out. However, due to some problems with existing implementations (that are CPU intensive and consequently cause poor performance), we decided not to include, at this stage of our study, DSR in our testbed. OLSR protocol [Ton] is an optimization for MANET of legacy linkstate protocols. The key point of the optimization is the *multipoint relay* (MPR). Each node identifies (among its neighbors) its MPRs. By flooding a message to its MPRs, a node is guaranteed that the message, when retransmitted by the MPRs, will be received by all its two-hop neighbors. Furthermore, when exchanging link-state routing information, a node lists only the connections to those neighbors that have selected it as MPR, i.e., its Multipoint Relay Selector set. The protocol selects bidirectional links for routing, hence avoiding packet transfer over unidirectional links. AODV is a reactive protocol that minimizes the number of route broadcasts by creating routes on-demand [PR99], as opposed to maintaining a complete list of routes as in OLSR. Route discovery is initiated on-demand, the route request is then forwarded by the source to the neighbors, and so on, until either the destination or an intermediate node with a fresh route to the destination, are located. Having in our test-bed one proactive and one reactive routing protocol enables us to compare these two approaches in a realistic scenario. In the literature, it is a common understanding to consider that on-demand reactive protocols are more efficient than proactive ones. On-demand protocols minimize control overhead and power consumption since routes are only established when required. By contrast, proactive protocols require periodic route updates to keep information current and consistent; in addition, maintaining multiple routes, which might never be needed, causes unnecessary routing overheads. On the other hand, proactive routing protocols provide better quality of service than on-demand protocols. As routing information is constantly updated in the proactive protocols, routes to every destination are always available and up-to-date, and hence end-to-end delay can be minimized. For on-demand protocols, the source node has to wait for the

route to be discovered before communication can happen. This latency in route discovery might be intolerable for real-time communications. One of the aim of our testbed is to compare and contrast AODV and OLSR from the efficiency and QoS standpoint.

**Middleware and Applications**

Solving MANET routing and forwarding issues is only a first step towards deployable MANETs. Integrating applications on top of an ad hoc network is fundamental to stimulate the users interest in this technology, and hence to better understand its possible usage scenarios. The middleware layer operates between the network layer and the distributed applications (i.e., it mainly implements layers 5-7 of the OSI model), with the aim to build on top of raw network services, higher level mechanisms that easy the development and deployment of applications. Mobile ad hoc systems currently developed adopt the approach of not having a middleware, but rather rely on each application to handle all the services it needs. This constitutes a major obstacle in the development of MANET applications. Ad hoc networks share many concepts with the peer-to-peer (p2p) systems [CGT04] (e.g., distribution and cooperation), hence p2p constitutes a natural computing model for ad hoc networks, as well. While research on middleware tailored for mobile ad hoc networks is still in its infancy, several p2p systems have been proposed and deployed for the Internet. p2p systems provide services ranging from file sharing to instant messaging to distributed and collaborative work. Thus, integrating p2p systems on top of ad hoc networks makes the variety of p2p applications and services available to MANET users as well, and hence it would be an advantage for this emerging technology. However, it is not clear how these overlays should be ported, and how they will perform on ad hoc networks. A defining characteristic of p2p systems is their ability to provide efficient, reliable, and resilient routing between their constituent nodes by forming structured self-organizing topologies on top of a real network infrastructure. In the p2p world, the requirement of a reliable way of partitioning the workload among all the participating peers and the observation that in a p2p system peers will be joining, failing, and leaving continuously, made self-organizing overlay networks (among peers) a promising approach to achieve the above goals with only a low communication overhead in the underlying network [D5]. This architectural separation

between routing at the network and middleware level, inspired a generation of *p2p routing substrates* (overlay network) which are based on a distributed message passing paradigm, where location of an object or a service is based on a key. The most popular approach adopts a Distributed Hash Table (DHT), in which nodes belonging to the routing substrate are assigned a unique pseudo-random identifier that determines their position in a key space. For example in Pastry [RD01] the key space has a ring structure. Messages are routed inside the overlay to points within the same key space and are delivered eventually to the node of the ring corresponding to the closest key. In other words, a message destined to a given key represents a request to provide a given service with respect to that key. Different variants of this basic approach exist, see for example [D5]. CAN, Chord and Pastry are examples of middleware that exploit DHT for constructing and maintaining p2p overlays.

In our testbed we wish to start understanding how these overlays perform on top of an ad hoc networks. Specifically, we focused on investigating the performance of Pastry on top of our ad hoc network. Results from this investigation provide the basis for defining efficient ways to port Pastry on ad hoc environments. Pastry differs from other p2p routing substrates such as CAN and Chord due to the structure of information on nodes and the way messages (or sometimes requests for routing information) are passed between peers. We selected Pastry for our testbed because it scales with the network size, and a free implementation (*FreePastry* [WSh]) with a rich set of services is available. Pastry works as a substrate for distributed p2p applications, offering the abstraction of a distributed hash table in which items can be located in a bounded number of routing hops, using a small per-node routing table. The main characteristics of Pastry, and its FreePastry implementation, are summarized below. The overlay network entirely founds its functionalities on the subject-based routing protocol, in order to uniformly distribute workload and data on nodes taking part to the same service. To this aim, middleware routing tables represent a building block of Pastry model. They consist of three data structures:

- ✧ *Leafset*. This is the set of logical neighbors of the local node, physically scattered on the network, representing a subset of the ring centred in the

local nodeID.

- ✧ *Neighborhood set*. This is the subset of the physical neighbors of the local node that are logically scattered on the ring.
- ✧ *Routing table*. It is used to define the next hop on the ring for a given message; it contains a predefined number of logical neighbors of the local node. Each row contains a set of IDs that share with the local nodeID a prefix long as the index of the row. In this way, a message is sent to the logical closest node related to the key value of the message, according to the proximity metric.

Since the logical address space is big enough to take in a large number of peers, each of them can maintain only a part of the Pastry network topology, depending on the routing table size and the logical address of the local node. For this reason the final destination of a message can be unknown to the sender, requiring a multi-hop routing at the middleware layer. The standard dimensions of the middleware routing tables are such that, in ad hoc networks consisting of a small number of nodes, all peers can know all the others, and thus it is not necessary to execute a multihop routing at the middleware layer. However, at the same time, operations needed to create and maintain the ring represent a high cost for ad hoc networks. When a node enters the Pastry network, and it is not the first one, it has to contact one of its physical neighbors already present in the ring in order to collect routing table information. The node contacted sends a message on the ring using the new node ID as the key of the message that thus will reach the node logically closest to the specified ID. At this point the new node can initialize its routing tables receiving the *Leafset* from the logical closest node, the *Neighborhood set* from the specified physical neighbor, and the *Routing table* as a join of the routing tables of nodes that forwarded the original message. Each of these operations requires several remote connections that are implemented with TCP connections in FreePastry and thus introduce a high cost. In addition, to maintain the overlay structure, each node has to execute a polling procedure needed to discover neighbors' status, so that a node is considered disconnected from the ring if it doesn't answer to a polling message before timeout expiration. If this occurs, the sender of the polling message has to update its routing tables contacting

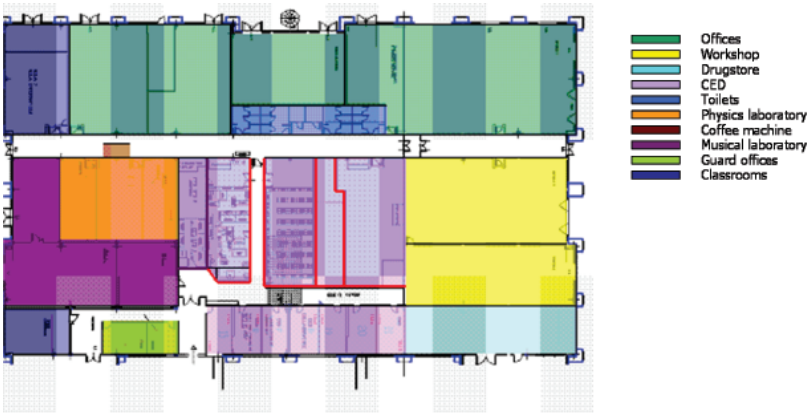


Figure III.2.: Experiments Area

other remote nodes, which thus implies opening other remote connections. FreePastry implements the polling procedure using UDP connections and routing table data exchange using TCP connections. In this way it is possible that a node has to maintain several TCP connections to different nodes only to manage the ring and this introduces a high overhead in ad hoc networks, mainly when topology updates are frequent.

## 3. Experiments Environment

All the tests were conducted at the ground floor in the CNR campus in Pisa (Figure III.2). At this level there is the computing center (CED) together with some companies' offices and measurement laboratories with several kinds of instrumentations. The structural characteristics of the building, and particularly of this floor, strictly determine the transmission capabilities for the nodes of a wireless network situated within. Rooms (offices, laboratories, etc.) are generally delimited by masonry padding walls situated between reinforced concrete pillars. In the CED area, instead, locations are separated by either *sandwich panels* of plastic materials, which don't reach the height of the ceiling, or by metal panels till the ceiling: these generally cause minor impediments to the waves compared to masonry walls

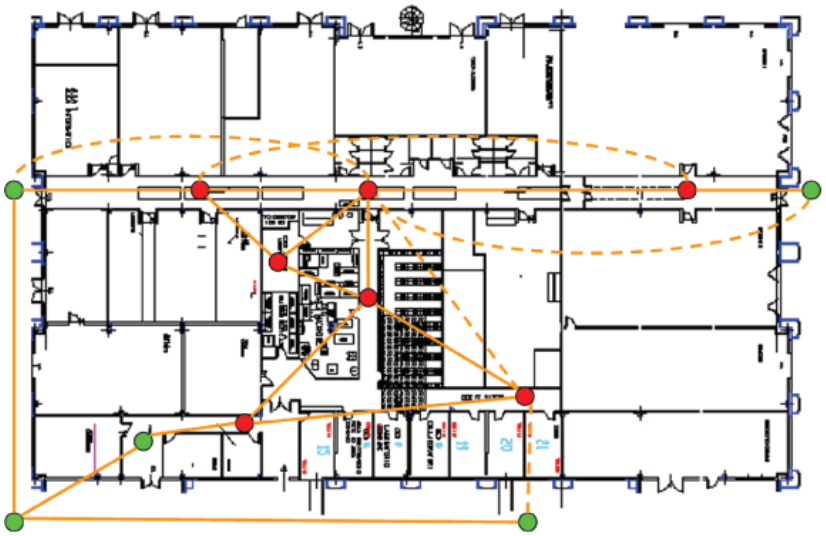

Figure III.3.: Experiments Scenario

or reinforced concrete pillars. Wireless links are also influenced by the presence in the nearby of Access Points and measurement instrumentation which introduce quite a lot of noise. Moreover, about $30 - 40$ people work in this floor every day and get around from office to office or towards service areas with coffee machines, toilets, etc. This makes the transmission coverage characteristics of the floor and the stability of the links varying in a contiuous and unpredictable manner. As a result the whole place can be considered quite a realistic environment for testing an ad hoc network. Figure III.3 presents the detailed map of the place together with the static transmission coverage characteristics of the area. Nodes are situated where devices were placed during the experiments and straight lines are used to point out the presence of wireless links (two nodes see each other at one hop distance if a single straight line joins them). Dashed lines are used instead to point out weaker wireless links wherever a couple of nodes see just sometimes each other and their communications are affected by a considerable packet loss. The devices used for the experiments were both laptops and PDAs (Compaq iPAQ $3950$) and had wireless cards with different capabilities: this caused some links to appear/disappear in different experiments depending on the power of the wireless

cards used by the nodes at each side of the link. Specifically, we used three different types of wireless cards: PCMCIA DLink $DCF-660W$ for PDAs, and D-Link DWL $650$ ($15$ dBm) and 3COM 3CRWE62092A ($14$ dBm) for laptops.

In the experiments performed we utilized a limited number of nodes ranging from 5 up to 12 nodes. These numbers may appear not meaningful respect to simulations scenarios using hundreds of mobile nodes. However, recent results pointed out the existence, with the current technology, of an ad hoc horizon of two-three hops and 10 to 20 nodes. Beyond these limits the benefit from wireless multihop ad hoc networking virtually vanishes [GLNT04]. Indeed all the experiments presented hereafter fall inside this ad hoc horizon. This may represent a scenario consisting of few people forming an ad hoc network to share documents. The focus of our study is therefore to contrast and compare ad hoc networking solutions *within* current technology limits. Currently, networks of hundreds of mobile nodes connected to several hops seem to be an unrealistic goal.

## 4. Experiments warm-up: a qualitative analysis

An extensive first experimentation phase was carried out during June $2004$ at the CNR (Italian National Research Council) of Pisa. The aim of the experimentation was to test if the software at each layer of the protocol stack behaves correctly. Specifically, the test concerned different layers:

- ✧ Routing: testing the selected implementations of proactive OLSR and reactive AODV routing protocols to check their state of implementation, validate their functionalities and conduct a comparative analysis on them all. The considered routing protocol implementations were the UNIKOLSR [Ton] by the University of Oslo (Norway), and the UU-AODV [WSm] by the Uppsala University (Sweden).

- ✧ Middleware: testing the selected FreePastry implementation on top of proactive and reactive routing protocols and evaluating the heaviness of the resulting solutions. The middleware platform used in these experiments was

FreePastry-1.3 which is the open-source implementation of the original Pastry model developed by the RICE University [WSh]. To this end we installed FreePastry on a set of laptops which had been previously equipped with the j2sdk-1.4.02 [WSo] Java Virtual Machine. FreePastry implements the p2p common API [DZD+03] that was proposed to guarantee the same interface between the applications and all the middleware platforms based on the overlay networks.

- ✧ Application: evaluating the impact of routing protocols on the Quality of Service (QoS) experienced by the application (e.g., the transmission delay). We used both legacy Internet applications (e.g., ping and ftp) working directly over the TCP/IP protocol stack, and a *Distributed Messaging* application running on top of FreePastry by exploiting the p2p common API.

Hereafter, we will report the main results of this experimental phase, a detailed presentation about all the experiments can be found in [D8]. The experiments were organized in two steps. First we tested OLSR and AODV in isolation to verify that they behave correctly. Then we integrated FreePastry to investigate the behavior of our full MANET.

## 4.1. UNIK-OLSR Testing

Due to the proactive nature of the protocol the test was based on observing the status of the nodes routing tables while nodes were added (removed) to (from) the ad hoc network. This testing was subdivided in two steps. In the first step we used a 5-node network. In this case the kernel routing tables were small and could be read in real-time, hence it was possible to follow configuration changes while in progress. Upon the beginning of the experiments, node insertions and removals were provided so to check that configuration updates effectively took place. Moreover, by changing the time lag duration between successive node insertions and/or deletions, it was also possible, to some extent, measure the configuration-update delays after the appearance/disappearance events. In all the experiments the protocol showed a correct behavior. The routing tables quickly updated upon node insertion and removal. We then considered a 12-node network. The increased

number of nodes led to the increase of the number of protocol packets exchanged. This allowed the validation of UNIK-OLSR behavior in a more congested context. Also in this set of experiments, all the routing-forwarding operations were correctly performed. After this analysis, we investigated the ability of an OLSR-based network to transfer data between nodes at a distance of a few hops. To this end, the UNIK-OLSR protocol was started on all the nodes at the same time, then after a little delay to let the routes stabilize, an FTP transfer was started between two nodes. The destination was at three hops distance from the source. The aim was to transfer a $34$ megabytes (MB) file. Several problems were experienced in this case. Intermediate nodes along the sender-destination path stopped working correctly after a while. This was due to the wireless card not properly working. It seemed that the excessive traffic they had to manage caused problems to their cards' drivers. In these experiments the routing protocol still behaved correctly by selecting alternative routes to avoid the out-of-service nodes. The file continued until a network partition occurred. At this time the destination host had received only the first $15$MB of the file. The throughput during the transmission had just been about 180Kbps. We repeated the experiment and observed similar problems. Specifically, we observed that the file-transfer started correctly but while the transfer proceeded the throughput of the connection reduced. This type of behavior can be explained with problems produced by the interaction between TCP and the $802.11$ MAC extensively investigated in the literature, see Chapter 3 in [BCG04] for a summary[2].

## 4.2. UU-AODV Testing

As this protocol is proactive, some application-level traffic was introduced in order to observe the route creation process. Specifically, each node sent periodically a set of pings to different destinations and this forced the routing protocol to set up, for each ping operation, a route toward the ping-destination node. In this case, each sender was always able to discover the correct paths towards the destinations

[2] Indeed, in these experiments we used default values for TCP parameters, e.g., advertised window. It is left for further studies to investigate the impact of TCP parameters on the ad hoc network performance.

chosen, however the discovery of the paths was very time-expensive. In the next section, some estimates of these delays will be provided. After this, we tested the UU-AODV ability to support user-data transfer. Again we used a file transfer application. The file transfer was started, from a couple of nodes at a 3-hop distance, with the aim to transfer a 5MB file. The transfer was definitely too slow and after 16 minutes only 140KB had reached the destination; the experiment was then interrupted. Again, the problem seemed related to interaction of MAC and TCP mechanisms. Packet losses caused a TCP congestion-reaction that slowed down the connection throughput. In addition, in this case, the reactive nature of the routing protocol made the things worse.

## 4.3. FreePastry test

The last set of experiments was carried out in order to evaluate the overhead introduced by a middleware platform based on the original Pastry model [RD01] and to validate its functionalities on the ad hoc networks. To test the environment we integrated on top of FreePastry a simple application of Distributed Messaging (DM), aimed at testing the main Pastry functionalities. Nodes participating to DM set up and maintain a Pastry ring corresponding to this Pastry service. Specifically, each instance of the messaging application defines an Identifier (ID) for the local node on which it runs. The ID is then used as the key for implementing the Pastry distributed hashing operations. When the application starts, the operations for the Pastry ring initialization depend on the existence, or not, of a Pastry ring already providing that service. In the first case the node performs the operation required to join the ring, otherwise, the node is the first of the ring, and creates a new ring. If a ring already exists, the user has to specify the IPaddress of a known physical neighbor so that the local node can connect to it, collect its middleware routing table and enter the ring. Once the local node has created/joined a ring, the application provides to the user the possibility to create a mailbox, to delete a mailbox on the nodes of the Pastry ring. A mailbox physical location is randomly selected applying the DHT to the identifier associated to the mailbox, i.e., the identifier represents the key of the message on which the hash function is computed. Once the mailboxes are created, the user can send/receive a message to all mailboxes.

These operations are based on the Pastry subject-based routing mechanism. Our DM application also generates 100 mailboxes to which it randomly associated IDs. Using this application we tested FreePastry to verify the overlay construction, the workload it generates, and the data distribution on the Pastry ring. Moreover it is also possible to evaluate the overhead introduced by this platform on ad hoc networks. These tests showed that all operations were performed correctly. However, the produced overhead was quite high, particularly, due to the large number of remote connections needed to maintain the overlay structure. This overhead will be quantified in the next section. The same set of experiments (i.e., the same network configuration) was performed by running UNIK-OLSR and UU-AODV. The network topology was kept as much similar as possible to the one used for previous experiments. In the network we had 8 nodes: 6 nodes ran FreePastry and the others 2 just worked as routers. During the experiments the nodes started running either UNIK-OLSR or UU-AODV then (after a delay of a few seconds to have the network topology stabilized) they ran the DM application trying to build a single Pastry-ring. From the performed experiments we noticed that rarely application messages had to execute a multi-hop middleware path. Hence the real overhead introduced by FreePastry on ad hoc networks is due to the periodical remote connections needed to the ring maintenance. In this set of experiments, the main problems were observed when running UU-AODV. In this case, we experienced a high number of Pastry connections' failures (i.e., the connection used by Pastry to maintain the ring). This was mainly caused by the reactive procedure to discover a route towards a specified node. When a node tries to connect to another one, FreePastry generates a TCP connection in order to recover middleware routing table information. If the local node has no routes to the destination, AODV generates a Route Request and waits for the answer. In the meanwhile, if path discovery is slow, the timeout of the Pastry connection may expire causing the raising of a Java exception. As TCP remote connections are periodically executed by FreePastry in order to maintain the overlay structure, the previous problem causes the wrongly notification of *death node*, even if it is still connected to the ring[3]. Therefore, using Pastry, the overhead reduction introduced by reactive protocols is cancelled by pe-

[3] The AODV implementation we used caches routes into the kernel routing table for 15sec only. Therefore, a new path discovery phase is generally required when Pastry performs its ring maintenance operations.

riodic remote connections. UNIK-OLSR experiments didn't suffer these problems thanks to the continuous update of the kernel routing tables.

### 4.4. Lessons

To summarize, our preliminary evaluation (mainly qualitative) of a full MANET prototype indicated that:

- ✧ UNIK-OLSR and UU-AODV behave correctly. They discover multihop paths and reconfigure the paths upon node failure/insertion.
- ✧ FreePastry operates correctly on top of the multi-hop ad hoc networks.
- ✧ TCP data transfers on top of a multi-hop ad hoc networks show severe performance problems. The performance problems seem to increase when using a reactive routing algorithm due to the delays caused by the reactive paths discovery.
- ✧ FreePastry overheads are mainly caused by ring creation and maintenance operations. These operations are based on the TCP protocol. As pointed out in the last point, TCP operations exhibit poorer performance by using AODV. In the FreePastry this causes problems in the correct execution of the ring maintenance operations, as TCP delays cause the timeout of the Pastry connection expire causing the raising of a '*Connection Refused* message at the Pastry level.

## 5. Quantitative analysis

A second set of experiments was performed, in the same environment used so far (see Section 3), to provide a quantitative estimation of the most interesting phenomena observed. Again, we first measured the performance of OLSR and AODV in isolation, then we analyzed the testbed integrating FreePastry on top of the ad hoc network.

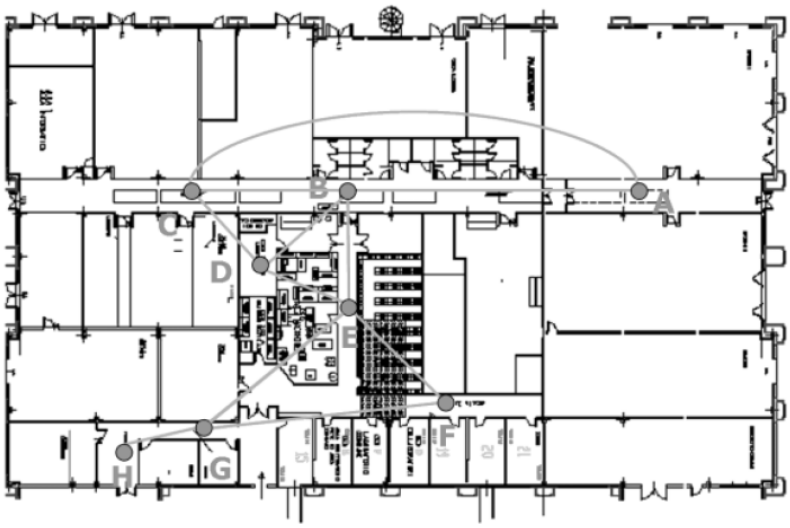


Figure III.4.: Routing Network Topology

## 5.1. AODV and OLSR Performance

The experiments were made in the environment shown in Figure III.3. For ease of reading, in Figure III.4 we report the same scenario in which we label the MANET nodes in order to identify them in the following discussion. The figure show the 8-node scenario on which the results reported below have been obtained. A line among a couple of nodes indicates that a link exists among them[4]. The aim of the experiments was to compare the two routing protocols in terms of:

- ✧ overhead introduced in the network due to the routing messages;
- ✧ the delay introduced for path discovery.

To have a meaningful comparison we used the ping application to generate some user-level traffic. Without any traffic, the reactive protocol (AODV) does not per-

[4] It can be noted that some links that were marked as unstable in Figure III.3 are now marked with solid lines to point out that they are now stable links. The reason for this is that we removed some obstacles to signal propagation (e.g., fire doors, that previoulsy caused weak links, in these experiments were opened).

form any activity. Two set of experiments were performed depending on the way the ping operation was performed.

### 5.1.1. Experiment 1

In this set of experiments, the central node E performed a ping operation towards the other nodes of the network according to a randomly selected sequence: A,H,D,F,G,B,C. For each node in this sequence, the node E performed the ping operation for 1 minute, then it moved to ping the next node in the sequence. The same sequence was used in all the experiment of this set. We performed several experiments that produced similar results. Hereafter, we present the results obtained from one of these experiments. These results are summarized in the Figure III.5 and FigureIII.6 for OLSR and AODV, respectively. In the figures we report the amount of traffic (expressed as number of Bytes per second) forwarded by each node of the network. More precisely, this traffic includes both the routing traffic generated by the node itself and the routing traffic generated by the other network nodes and forwarded by this node. As expected, the traffic depends on the part of the network a node is located. By looking in details to Figure III.5 we can note that:

- ✧ nodes B and D (the two upper curves) have to forward the highest quantity of traffic (about $1.1$ KBps);
- ✧ nodes C, E, and G (the curves around $0.8$ KBps) have an intermediate load;
- ✧ nodes A and F have a $0.4$ KBps load.
- ✧ node H is lightly loaded (its traffic, $300$ KBps in average, is about 1/4 of node B and H traffic).

This traffic partitioning shows a good correspondence with the role of the nodes inside the network. H is a leaf in the network graph and it has only one link with node G. Nodes F and A are leaves, as well but they are connected to the network via two links. The other nodes are inside the core of the network and have several neighbors thus their traffic is higher. It is interesting to note that, in the AODV

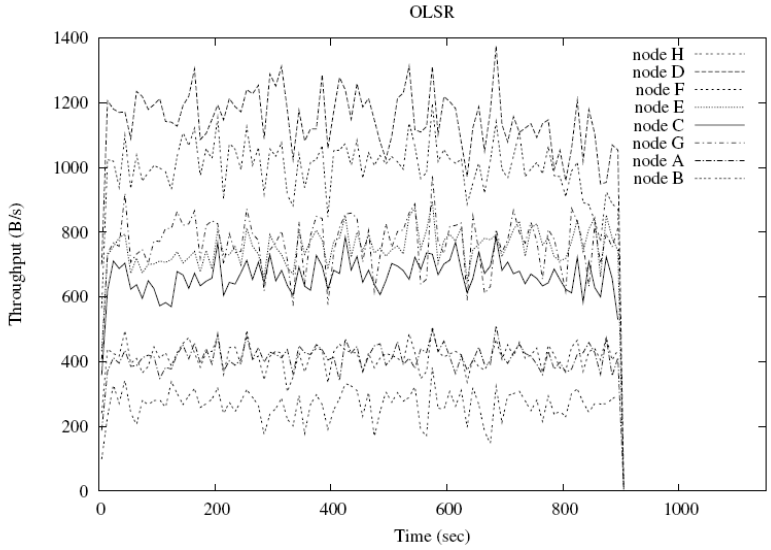


Figure III.5.: OLSR overhead

case these relationships between a node position and the traffic it observes do not always hold. For example, see Figure III.6, while node H still experienced the lowest load, node B (which has the highest traffic with OLSR) had an intermediate load. This is due to the reactive nature of the AODV protocol that makes the routing traffic dependent on the traffic flows at the application level. As expected, OLSR generated a traffic that was significantly higher than that produced by AODV. Specifically, while with AODV the traffic range was $[100-400]$ bytes/sec, with OLSR was $[200 - 1200]$ bytes/sec. However, even though in percentage this difference is big, it is important to note that in both cases the impact on the utilization of the 802.11b bandwidth is almost negligible. A node observed at most $1.2$ KBps of routing traffic. On the other hand, delay measurements pointed out possible severe problems on QoS when using AODV. Specifically, with AODV, if the route was not yet in the nodes cache we measured, for completing a simple pingoperation between a couple of nodes at a 2-hop distance, delays in the order of $19 - 20$ seconds. To perform the same operation, OLSR requires $1$ second (or less) as routing tables are generally updated.

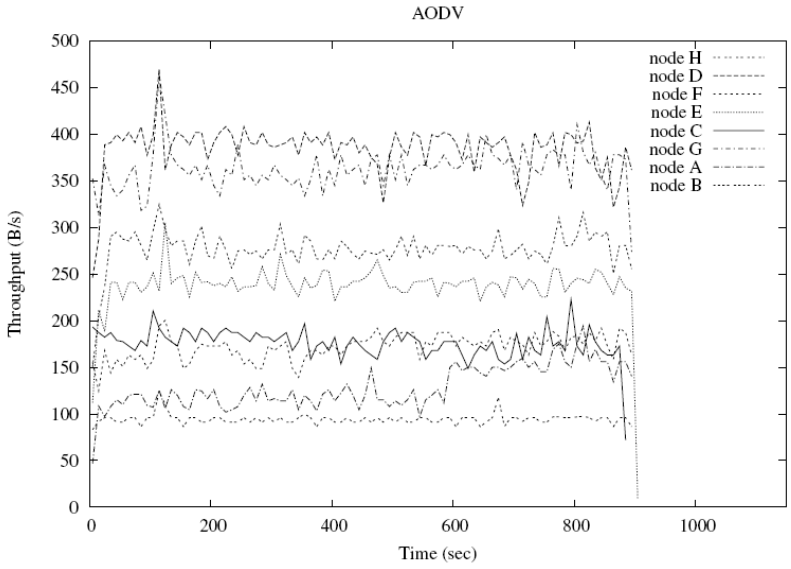


Figure III.6.: AODV overhead

### 5.1.2. Experiment 2

In this set of experiments, the external node H continuously (for $400$ seconds) performed the ping operation towards the node A. In this case the shortest path is: H-G-E-B-A. After $x$ seconds ($x$ equal to $250$ and $180$ seconds in OLSR and AODV experiments, respectively) from the beginning of the experiment, the node B disconnects from the network and this cause a change in the route from H to A that now is crossing D and C, instead of B. Again, we performed several experiments. Hereafter we report, as an example, the results from one of them. They are summarized in Figure III.7 and Figure III.8 for OLSR and AODV, respectively. As far as OLSR, we can note a load distribution similar to that observed in the Experiment 1, e.g., node B and node H experienced the highest and lowest load, respectively. Differences (in terms of amount of traffic) between Experiment $1$ and $2$ can be explained by the different applicationlevel traffic flows. After node B shut down, there was a transient phase during which the node-traffic decreased (due to some missing routes), after which a new steady state was achieved. In this

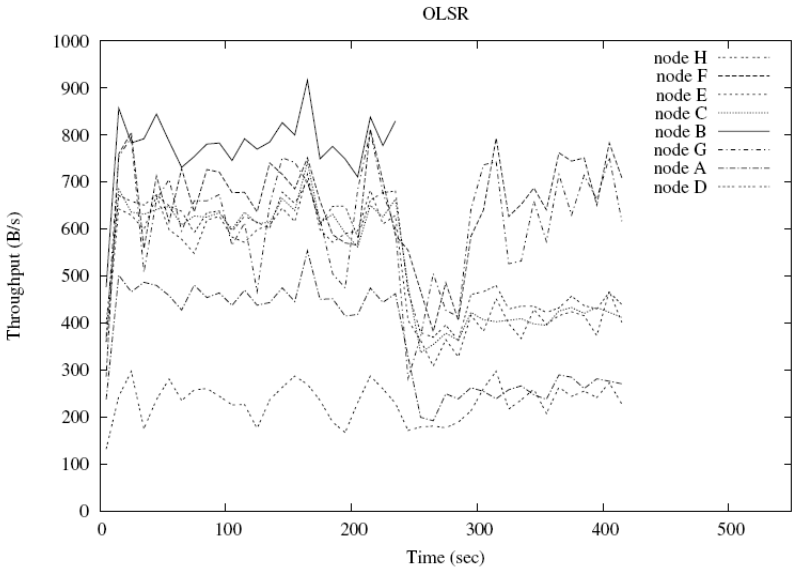


Figure III.7.: OLSR overhead in case of disconnection event

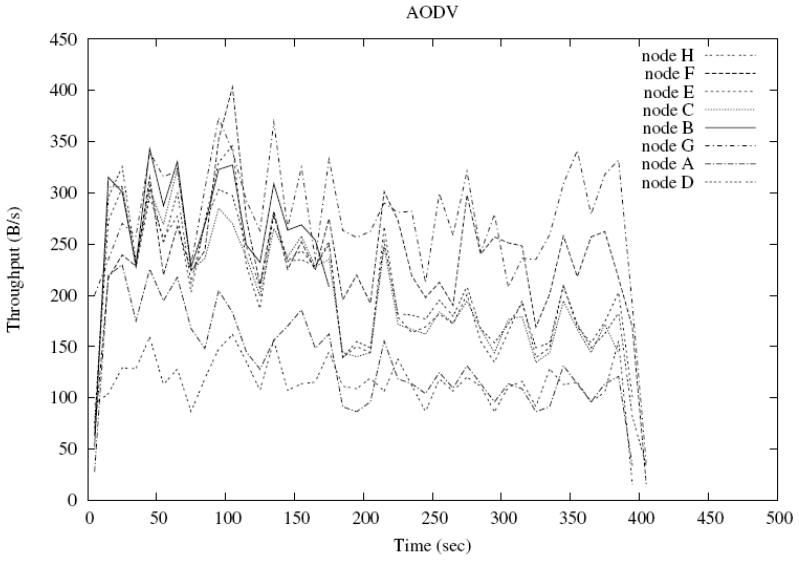


Figure III.8.: AODV overhead in case of disconnection event

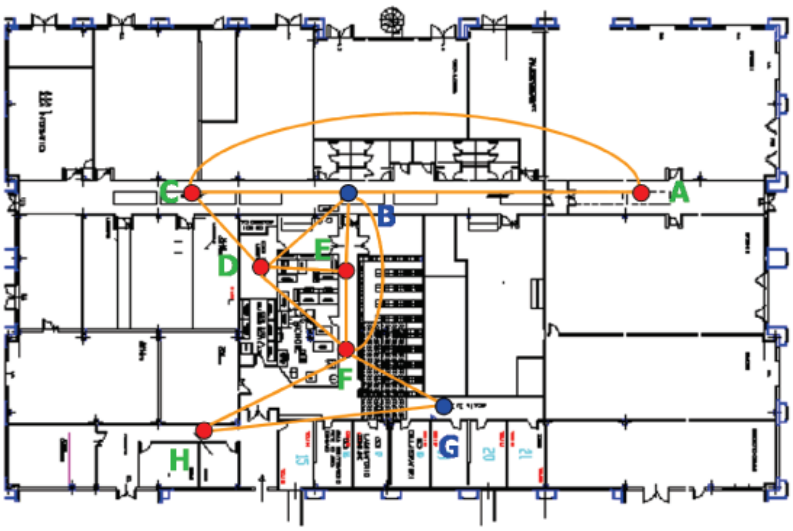


Figure III.9.: Network Topology for FreePastry Experiments

new steady state, we observed a significant decrease of the traffic in the nodes that were connected with node B (A, E, D, C). Less marked differences, before and after B shut down, can be observed by looking at AODV results. After the transient state following B shut down, the active nodes almost observed the same load experienced before the shut down event. As far as the delay is concerned, we got results similar to those observed in Experiment 1: very large delays with AODV (about 20 seconds) when the path is not already in the cache. On the other hand, OLSR introduced very small delays. The only exception was after B shut down which caused a routing tables reconfiguration. In this case the ping operation performed while OLSR was updating the routing tables experienced a delay of about 6 seconds to be completed.

## 5.2. Performance of FreePastry on Ad Hoc Networks

This set of experiments was made by adopting a network topology (see Figure III.9) which slightly differs from that used for the analysis of the routing algo-

rithms. We decided this modifications as experiments shown in Section 4 indicated that central nodes in the network tend to become saturated. To avoid this, we moved one node towards the center of the network. In this way we increased the redundancy in this area. Among the eight nodes of the ad hoc network, 6 provided the Pastry service, while nodes B and G (the empty circles in the figure) were only involved in routing and forwarding operations. As far as the Pastry operations, node E was the first to start and then it initialized the overlay ring related to the distributed messaging service. Then the following actions were performed in sequence: F joined the ring by connecting to E, D connected to E, C connected to D, A to C, and last H connected to F. At this point all the nodes providing the Pastry services were connected to the overlay ring. In the next figures we investigate the costs (in terms of traffic) required to maintain the Pastry overlay ring on top of our ad hoc network. It is worth remember that Pastry generates management traffic (to maintain the overlay ring) both when a node join the network (and hence it needs to acquire the information about the other nodes belonging to the same service), and periodically to check the status of the overlay ring. The latter operation performed by each node is implemented by opening TCP connections from that node towards all the other nodes belonging to the ring.

Figure III.10 and Figure III.11 present the amount of traffic forwarded (including their own generated traffic) by the network nodes using OLSR and AODV, respectively. As expected, node B and G that only participate to the routing and forwarding operations generate the same amount of traffic observed in Section 5.1 in which the traffic was mainly due to routing operations. On the other hand, the nodes belonging to the Pastry ring periodically experience a burst of traffic produced by the overlay-ring maintenance operations. To better investigate this aspect, in the following graphs we focus on a single node and analyze the type of traffic it generates. Specifically, we identify four traffic classes:

- ✧ the traffic due to the ARP-protocol operations;
- ✧ the routing traffic;
- ✧ the UDP traffic generated by Pastry ring maintenance operations, e.g., ping operations performed at Pastry level by a node to test the other nodes in the ring;

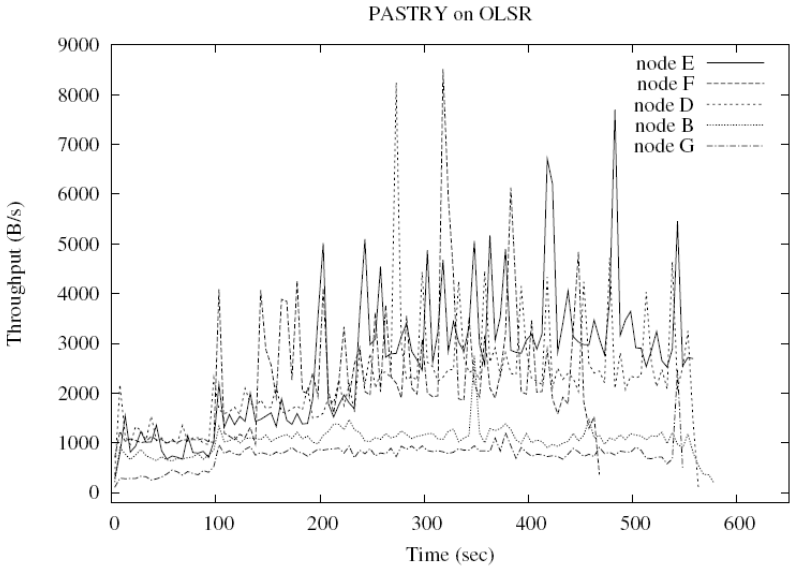


Figure III.10.: Traffic Load on each node running Pastry on top of OLSR

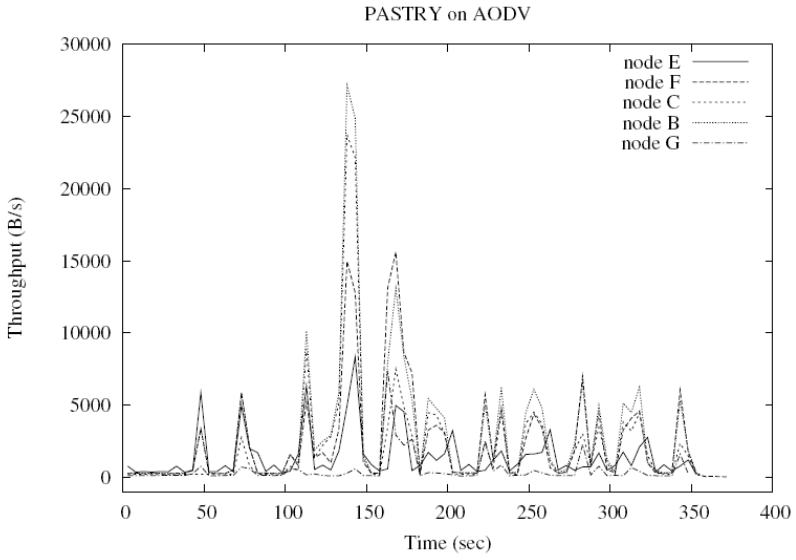


Figure III.11.: Traffic Load on each node running Pastry on top of AODV

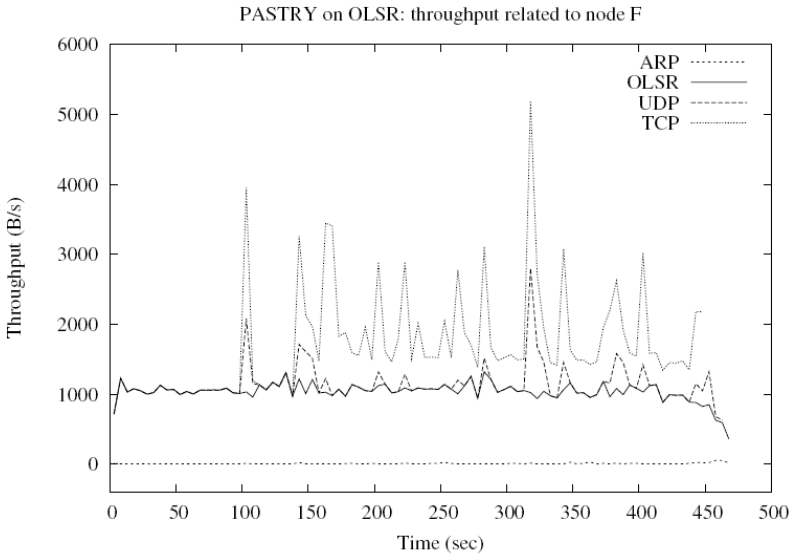


Figure III.12.: Node F traffic on an OLSR network

- ✧ the TCP traffic used by the nodes belonging to the Pastry ring to exchange their routing tables and other ring-related information.

In the figures below we report the total traffic generated by a node. More precisely, while the curve ARP indicates only the total traffic produced by ARP protocol, the curve OLSR indicates the sum of ARP + OLSR traffic. The curve UDP denotes the sum of ARP+OLSR+UDP traffic, and the TCP curve is the node total traffic (i.e., ARP+OLSR+UDP+TCP).

Figure III.12 and Figure clearly pointed out that the traffic peacks are due to the overlay ring management (i.e., TCP and UDP traffic). Indeed, from these figures we can observe that the ARP traffic is almost negligible, while routing traffic is quite regular and provides links utilization levels similar to those observed in Section 5.1 without FreePastry. On the other hand, the traffic burstiness is almost only due to TPC/UDP traffic required by the overlay ring maintenance operations[5]. These observations are confirmed by observing other network nodes. In

[5] The flat behavior in the first 100 seconds of the OLSR figure was due to our choice to start the Pastry

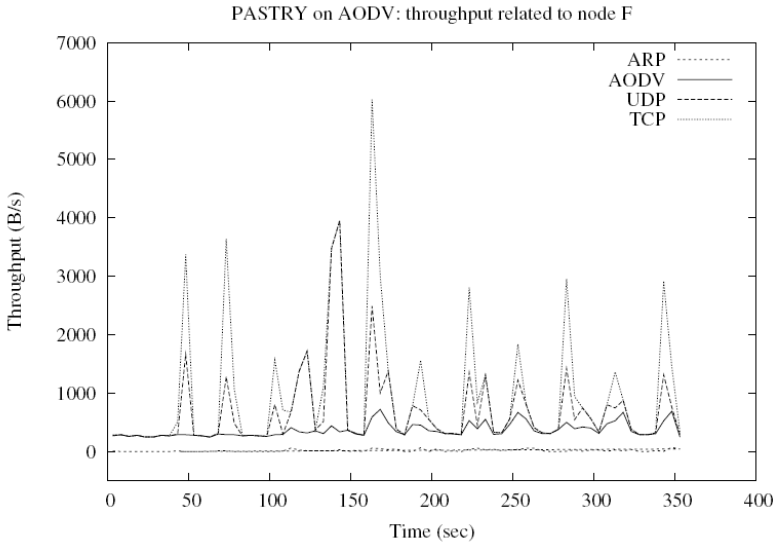


Figure III.13.: Node F traffic on an AODV network

particular, by observing node A, when using AODV (see Figure III.14), it clearly appears that Pastry operations may produce big traffic peacks, mainly during ring initializations.

## 6. Lessons learned

From our experimentation we can conclude that good pieces of software exist, correctly implementing the single functionalities required in a MANET. The implementation of AODV and OLSR we tested are quite robust. They are able to maintain updated routing tables even under frequent changes to the network topology. However, their usage is not yet user friendly. Problems were experienced depending on the release of the LINUX kernel. The FreePastry implementation we tested operated properly on top of our multi hop ad hoc networks. On

ring only after a 100 seconds delay in order to have the OLSR routing tables initialized. We did not add any delay in the AODV network because when no upper layer traffic is generated AODV does not add any information in the routing tables.

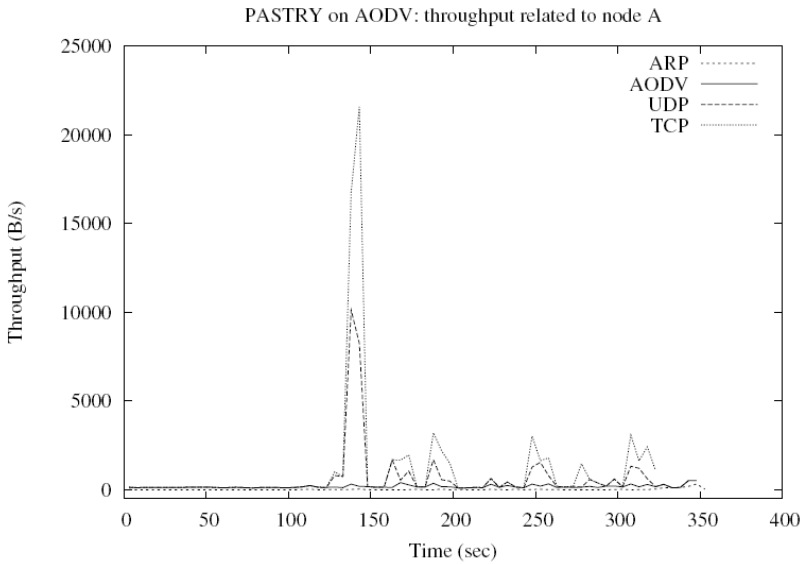


Figure III.14.: Node A traffic profile (AODV case)

the other hand severe problems have been identified from the performance standpoint. The problems affected almost all MANET layers: network interface, routing and forwarding, TCP, and Pastry. The quality of the wireless links is highly variable. IEEE 802.11 operates in the ISM spectrum and hence experienced a lot of noise from external sources. In addition, the increasing success of WiFi hot spots tends to saturate all the available channels. In our experimentation we have often to switch our MANET on a different 802.11 channel to avoid the influence of existing WiFi access points. Routing and forwarding performance problems were experienced when using AODV and are due to the reactive nature of the protocol. The delays due to path discovery and maintenance have a strong negative impact on upper layer protocols that use *connection-oriented* operations. Specifically, with AODV, when no route is in the node cache, we measured, for completing a simple ping-operation between a couple of nodes at a 2-hop distance, a delay of about 20 seconds (values of this order were experienced several times). For the same operation, OLSR takes 1 second (or less) generally having routing tables updated. In rare cases in which the ping operation was performed just after a change in the

topology and hence no updated route was available, we experienced a delay of up to 6 seconds to complete the ping operation. AODV path-discovery delays have a negative impact on upper layer operations. These delays often caused timers expiration in upper layer protocols (e.g., the FreePastry timeout related to ring maintenance) that, as a consequence, declared failed an operation that is indeed only delayed due to AODV delays. From the overhead standpoint we observed that, as expected, OLSR produces an higher routing traffic respect to AODV, but at least in the network we analyzed, the percentage of this traffic is small compared to the $802.11b$ available bandwidth. At the transport level, we experienced the TCP problems already pointed out in the literature. Long TCP connections show a throughput that decreases with time. This aspect requires further investigation in future experimentation to verify if an appropriate tuning of the protocol parameters is needed (e.g., advertised window). Finally, at middleware layer, the FreePastry implementation, by operating its own routing ring independently from the underlying ad hoc network, introduces a heavy overhead on the ad hoc network. In addition the FreePastry maintenance operations exploit the TCP services. Hence poor TCP performances couple with the FreePastry overhead to reduce the overall system performance. From this experience we gained some indications for solving the performance problems in our MANET. Specifically, these results provide additional arguments to using for a MANET a cross layer architecture as proposed in ([CMTG04], [CCMT04]). Results related to the comparison between reactive and proactive routing protocols indicate that, with a proactive protocol: i) the response times are much better, and ii) the protocol overheads, at least inside the ad hoc horizon, are not heavy. Furthermore, results related to FreePastry indicate that significant performance benefits can be expected if routing information (extended with services information) can be used at the middleware layer to implement the overlay ring maintenance operations. This way the big overheads connected with all remote connections are avoided.

# IV. From MANETs to Opportunistic Networks

## 1. Introduction

The last few years have assisted to the deployment of Mobile Ad hoc NETworks (MANETs) in many application environments. Originally conceived for military applications and aimed at improving battlefield communications and survivability, MANETs have lately begun pervading many civil scenarios. *Precision agriculture*, for example, makes use of sensor ad hoc networks to collect information about soil conditions and spatial and temporal variability in crop yield and quality. Data collected is exploited to fine-tune seeding, fertilizer, chemical and water use, thus potential increase production, lower costs, and reduce pollution. Sensor ad hoc networks are also used by biologists for *wildlife tracking*, i.e., to gather biological and position information from wild species like zebras, whales, and penguins, to study their habits, mobility patterns, migratory phenomena, etc. *Intelligent highways* are being developed by exploiting the possibility of ad hoc vehicle-to-vehicle and vehicle-to-roadside communications and provide driving assistance, improve driving safety (e.g., with emergency warnings) and comfort, and also allow some leisure activities (e.g., with interactivity games for passengers). Personal Area Networks (PANs) are used for *health care* applications. Specifically, by distributing biomedical sensor nodes on the human body, continuous monitoring of vital functions is possible that may help prevent acute crisis in healthy subjects, give assistance to the elderly, or even avoid hospitalization for long time after an operation. Ad hoc networks may also serve small communities of colleges or conferences for example, to provide temporarily services like *data sharing*. Finally,

*home automation* is emerging as a new interesting field of application. Lately supported by the development of the Zigbee technology, it allows low data rate and low power consumption exchanges between home appliances and devices and enables control systems like light and climate control, computer control and control of doors and window shutters. It also supports security and surveillance systems, multi-media home entertainment systems, automatic plant watering and pet feeding, automated blinds and curtains, automatic scenes for dinners and parties, etc. As the application environments of MANETs increase, their traditional communication paradigms need adequacy. Specifically, today's ad hoc networks are evolving towards *opportunistic networks* where the proactive and reactive approaches for routing management are integrated, or definitely substituted, with techniques that exploit *communication opportunities*, whenever they arise, to forward messages on a hop-by-hop basis. In fact, in opportunistic networking no assumption is made on the existence of a complete path between couple of nodes wishing to communicate. This well suits many of the new application scenarios because intermittent connectivity is likely to occur. As a consequence, efforts are made to look for *partial* routes ending at relay nodes where the forwarded messages can be stored until a next hop is available again towards a new relay or the destination itself. Most of the concepts behind opportunistic networks come from the early studies on *Interplanetary Networks (IPNs)* and, later, on *Delay Tolerant Networks (DTNs)*. Although these kinds of networks are not properly ad hoc networks (especially the IPNs), they primarily face the problem of intermittent connectivity and make use of techniques based on *store and forward* for routing. IPNs and DTNs consist of networks of internets each characterized by Internet-like connectivity in within, but having only occasional time-disjoint communication opportunities among them, sometimes scheduled over time, some other completely random. This chapter offers an overview on the Interplanetary and Delay Tolerant Networks then specifically focuses on opportunistic ad hoc networks starting from their theoretical foundation, going further with some recent experimental studies and projects, and ending with the state of the art of their routing strategies and techniques.

# 2. From InterPlaNetary networks (IPNs) to Delay Tolerant Networks (DTNs)

In the early 1998, the DARPA Next Generation Internet initiative funded the InterPlaNetary Internet Project [CBH+01] [CBH+03] [AAC+03] with the aim to define a communication system together with Internet-like services across interplanetary distances. The project was originally conceived to support deep space exploration. However, the vision of a *stable interplanetary backbone network* among planets, satellites, asteroids, robotic spacecrafts, and possible crewed vehicles later gave rise to other new challenging applications from the mining of asteroids to the deployment of space-based hotels for tourism support. During the project design phase, soon came out that the terrestrial communication systems are ill suited for the deep space environment because they rely on assumptions that are not verified in the deep space, such as:

1. *End-to-end connectivity:* a complete path between the source node and the destination node is expected to exist when the transmission starts.
2. *High capacity links:* terrestrial communications allow transmission of light-speed signals with short propagation delays (fractions of a second per single packet when using the TCP protocol).
3. *Reliable links:* terrestrial links have low bit error rate, even though they can potentially suffer from congestion due to the high demand for transmission.
4. *Symmetric links:* propagation delays from source to destination and from destination to source are generally the same.

Thank to the above assumptions, terrestrial communications can make use of *chatty* protocols to manage information exchanges (both data and control) between the communicating parties. Chatty protocols guarantee reliability and high reactiveness to data loss but, at the same time, make use of a communication model which is phone-call style because it needs near real-time exchanges to be performed. In the deep space environment, communications are instead characterized by:

1. *High propagation delays and round trip times:* distances between planets are enormous and lead to propagation delays of tens of minutes of magnitude (the round trip time at the speed of light, between Earth and Mars, for example, ranges between $8$ and $40$ minutes). This makes unfeasible utilization of transport protocols that rely on long handshakings between peer nodes (like for example TCP) because they would take too long to complete a single data transfer.

2. *Low data rates:* the radio signal easily degrades and attenuates over long distances.

3. *Intermittent connectivity:* a complete path between the communicating parties is likely not to be available due to the movements of the planets, the occultation of satellites during planet-orbiting, etc. In addition, distances between planets change over time and cause variations in delay, transmission capacity, connectivity, and topology. Intermittent connectivity results in long periods of *network partitioning*, and discontinuities in the capabilities of adjacent networks.

4. *Asymmetric links:* transmissions from Mars to Earth, for example, may be received at $100$ kilobits/second while Mars-based systems may only receive from Earth at $1$ kilobit/second.

5. *High bit error rate links:* links are definively error-prone.

6. *Low available bandwidth:* data rates range from hundreds of kilobits per second to few megabits per second and will probably remain unmodified in the next few decades. This is due to the combined effect of large distances, expense and difficulty in deploying large antennas to distant planets, and difficulty in generating power in space.

7. *Need for special equipment:* transmissions are generally very expensive over the deep space because they need special instrumentation like large antennas, high-performance receivers, etc. This further worsens the end-to-end delay experienced during transmissions because of the sharing of Earth-based resources that introduces scheduling and queuing delays. An efficient

use of the communication channel allows more information to be carried per unit of transmitted power.

It clearly appears from the above characteristics that *non-chatty* communication protocols, which make use of paradigms similar to the Postal and Pony Express systems'ones, are better suited for the deep space environment than chatty protocols. In addition, more suitable protocols should also *pack as much data as possible per single transmission* to minimize the number of round-trips needed. So, for example, a protocol for file transfers should send an entire file with the relative control data all together in a single *atomic* transaction. The total transmission would thus end faster within the IPN and make more efficient use of terrestrial high-speed networks. The architecture emerging from the InterPlaNetary Internet project consists of a network of internets, i.e., a collection of independent internets eventually interconnected by a system of *IPN gateways*. Single internets are located apart from each other (e.g., on the surface of planets or satellites, or in spacecrafts) forming distinct *IPN regions*. Internets are independent each other in that each one has its own protocols to rely on. Protocols are chosen to best suit the particular infrastructure, communication means, and technologies available in the particular internet's region and they may differ from the protocols of the other IPN regions' internets. A novel *overlay protocol* is added on top of the traditional transport layers at all the IPN nodes to manage the end-to-end data transfers among the IPN regions. Two nodes that are adjacent in the overlay space may be many hops apart in the context of the underlying network topology. To identify a node belonging to an IPN network, a special address is introduced which consists of a *tuple* of two identifiers. The first identifier is the *region ID* and identifies, at the overlay layer, the IPN region where the node is located. The second identifier is instead the *regional identifier* and identifies the node inside its IPN region at the transport layer. It should be noted that the naming system may differ from IPN region to IPN region and, as a consequence, the regional identifiers cannot be resolved everywhere but in their own IPN regions. Message forwarding is thus performed in two distinct steps. Firstly, the destination region ID of the message is interpreted and the message is forwarded towards the appropriate destination IPN region (each node is able to interpret the region ID). Once arrived to the destination IPN region, the destination regional identifier is interpreted and the message forwarded

towards the specific destination node. Hence, message forwarding is managed at the overlay layer until the destination IPN region is reached, then at the transport layer where the regional identifier is interpreted and the message finally sent to the destination node. This modus operandi is referred to as *late binding*. Nodes can be discriminated in *hosts*, *routers*, and *gateways*. Host nodes may send and/or receive messages but they cannot forward them. Routers forward messages within the same IPN region and may optionally act as hosts. Gateways, finally, are able to forward messages between two or more IPN regions and are therefore the most complex nodes in the overall IPN architecture. Located at the boundaries between two or more IPN regions, gateway nodes implement as many protocol stacks as the number of IPN regions they belong to, and provide lower-layer protocol *conversions* whenever they relay messages from one IPN region to another. In fact, when communications occur inside a single IPN region they only need management at the transport layer which is provided by the specific transport protocol implemented in that region. On the other hand, communications between IPN regions follow, in sequence, the transport protocols of each single IPN region they traverse (they generally differ each other). Therefore, communications between IPN regions cannot be managed anywhere but at the overlay layer because this spans all the traversed IPN regions. Each transport connection is ended at the edge of the corresponding IPN region, at an IPN gateway. Then the gateway provides relaying the incoming data from one interface to another after having changed the transport protocol and connection. Also gateway nodes may optionally act as hosts. The overlay layer is a sort of long-haul transport protocol and is also called *bundle layer*. It is common to all the IPN regions and does not need translation at the gateways. *Bundle* is also the name of a message exchanged in an InterPlanetary Network. In fact, it collects many pieces of information including the user data generated by the source application and some control information also provided by the source application for the destination application. Additional information is about how to process, store, dispose of, and generically handle the user data. A bundle may also contain information for autentication at the destination host (e.g., login, password) as well as specifications for reliability, quality of service, security or even error recovery management. The motivation behind bundles is the need to reduce the total transfer delay while interplanetary distances

tend to keep it high. In fact, it is more effective to send all the necessary data at once to sustain processing at the destination side rather than in a long handshaking sequence. A bundle also includes a special header added by the bundle layer itself. The total length of a bundle can be arbitrary. However, some form of fragmentation is also managed at the bundle layer. During bundle transmission along the path between the source node and the destination node, it may happen that a next hop is not available for forwarding, i.e., no connection is available to the next hop (*intermittent connectivity*). This may happen because the resources to be used for transmission are temporarily busy in transmitting high priority traffic or because the next hop is only reachable through a scheduled connection (e.g., when a satellite is visible). Bundles therefore need for *buffering* while waiting for forwarding. Moreover, since the connection may be available in hours, days, or even weeks, *persistent storage* is typically required[1]. This is different from what generally happens in the Internet where messages are stored at intermediate routers for very short time, i.e., fractions of seconds. The *store and forward* communication paradigm is also exploited to allow minimization of interactivity between endpeers during a bundle transfer. So, the node that first sends a bundle keeps a copy of it in its persistent memory after transmission until receipt of an acknowledgement from the next hop. It is the first *custodian of the bundle* because it keeps the only reliable copy of the bundle and will use it for retransmissions, if necessary. When the bundle custodian sends the bundle to its next hop node, it requests the *custody transfer* for that bundle and starts a time-to-acknowledgement retransmission timer. If the bundle layer of the next hop accepts custody of the bundle, it returns an acknowledgement to the sender. If the bundle custodian receives no acknowledgement before the sender's time-to-aknowledgement expires, it transmits the bundle again to the next hop. The bundle custodian stores the bundle until either another node accepts custody of the bundle or the bundle time-to-live expires. Then, either a new bundle custodian exists or the bundle has no more reason to exist. Thus, the copy of the bundle is discarded by the (obsolete) bundle custodian. The time-to-live of a bundle is obviously much longer than the time-to-acknowledgement of any custodian. Node-to-node reliability is perceived

[1] It is important to note that communications may even be deferred when the connection to the next hop actually holds but a substantially more attractive path is expected to become available soon.

by local retransmissions managed at the transport layer inside single IPN regions. This is different from what happens in the TCP protocol where retransmissions are handled end-to-end and the source node retransmits the data in case of a missed acknowledgement from the destination node. This approach is unfeasible in an IPN environment because of the high end-to-end delays involved, whereas retransmissions inside single IPN regions are faster and more efficient. So, reliable transport layer protocols and custody transfers are used by the bundle layer to move points of retransmissions progressively forward towards the destination node. In case the source node desires final notification of delivery, the destination node should send a separate *return receipt* to the source after receiving the bundle. The return receipt is transmitted as a new bundle, and is subject to the same custody transfers as the original transmission. The return receipt is similar to those utilized within the postal system. In October 2002 a new Research Group was chartered within the Internet Research Task Force to generalize the InterPlaNetary Internet architecture to the *Delay Tolerant Network (DTN)* architecture [War03] [CBH+02] [CBH+05] [CBH+03]. This more general architecture is envisioned to become more and more important to terrestrial communications since the global Internet scenario is rapidly evolving with the introduction of new accessing technologies and the diffusion of new heterogeneous devices with different portabilities and also capabilities in terms of processing, storage, and energy supply. Internet is ever-more similar to an aggregation of *islands ("regions") of homogeneity* where the end-to-end data path between a source node and a destination node spans a wide variety of environments. Each single data hop is inside an area of homogeneity and is traversed nearly istantaneously whereas some delay is introduced between consecutive areas to change technology and protocol. Given the above similarities between the IPN architecture and today's Internet, the DTN protocol concepts are thought to have the potential to significantly enhance the Internet's ability to accommodate the extreme heterogeneity that is emerging, thus eventually supporting end-to-end delivery of information. Examples of terrestrial application environments for the DTN architecture are: Military Tactical Networks, Sensor Networks, and Intelligent Highways. In military tactical networks, nodes are interconnected by the means of helicopters, satellites, aircrafts, etc. Due to the high mobility of these communication resources, connections are

likely to be intermittent and sometimes asymmetric. Moreover, delays within connected portions of the network may be extremely variable. Sensor networks, on the other hand, are likely to suffer from partitioning because their nodes have energy constraints and periodically need to switch to a low power consuming state. In Intelligent Highways communicating nodes are on vehicles in the highways and data is routed from vehicle to vehicle until an infrastructure is reached. This kind of network is conceived for traffic congestion avoidance, calculation of alternative routes, localization of points-of-interest, safety care, etc. Another interesting application scenario for the DTN architecture has been recently investigated in the framework of the **Saami Network Connectivity (SNC) project** [DUP02]. The project has been aimed at providing network connectivity to the nomadic Saami population of the Reindeer Herders. Saami herders live across the Sápmi region (also known as Lapland) in the northest part of Sweden, Norway, and Finland. The *People of the Eight Seasons*, this is how Saami are named, move from their villages through the year following the migration of reindeers. Although there is great interest in protecting and defending habits, culture and traditions of this aboriginal population, there is also an increasing need for their integration into the modern society of their countries. In fact, since the income from reindeer herding is no longer sufficient to sustain the livelihood of herders, even more Saami people are taking up jobs not involved in herding like for example teaching, nursing, and journalism. The possibility for distance work and net-based business could allow Saami to continue to live according to their traditions and, at the same time, have much economic sustain. Network based services could also allow Saami children receive their education without the need to leave their parents to attend boarding schools. On the other hand, network connectivity could help support Saami rights to be more visible and influence the political and economical affair of their country. In its initial stage, the SNC project has only focused on providing email, file transfer, and cached web services to the Saami people. Reindeer herd telemetry is also going to be provided to support the herding activity itself. Even though referring to a very specific case of application, IPNs and DTNs have been the defacto starting point for opportunistic networking. In fact, the basic concepts of opportunistic networking are all enclosed in the IPNs' and DTNs' paradigms, from the intermittent connectivity to the store and forward communication model and

to the custody transfer technique. From here on these notes will only focus on opportunistic networking when applied to ad hoc and sensor networks, and will investigate the specific aspects that discriminate these networks from the ones described above.

## 3. Theory and case studies on Opportunistic Ad Hoc networks

Ad hoc and sensor networks, both sparse and dense, may take advantage of alternative routing strategies like those defined for the DTN architecture. Source and destination nodes, in fact, may be disconnected due to the mobility of nodes or because nodes are in the sleep mode to save energy. In this case both proactive and reactive routings fail and cause the transmission rejection. However, if the application can suffer some delay, it is possible to defer transmission to a subsequent moment and provide temporary storage of the transmission content in the meanwhile. Deferring transmission may also be appropriate when there is low bandwidth available in the network so as, adding extra transmission load may lead to congestion. Messages can be stored until a (partial) path to the destination is available thank to node mobility or because some nodes have woken up and, as a result, have increased network connectivity. The storage duration in opportunistic networking is typically longer than in classical routing approaches because nodes have to wait for a communication opportunity to occur. The mobility of nodes helps create communication opportunities and reduce the storage duration and the end-to-end delay of transmissions. An *arising communication opportunity*, when talking about opportunistic networks, does not exclusively refer to an end-to-end path between the source node and the destination node of a message. Rather, it may simply refer to the possibility to transmit to an intermediate node which is nearer to the destination node with respect to the source node. Transmissions are thus managed in a hop-by-hop fashion. Basically, at each hop, the best is done to find a route towards the destination node or at least to near the destination node in case it is not soon reachable or there is the risk of congestion. The term *contact* is also used to refer to a communication opportunity between two nodes. It may consist of both a single hop connection or of a multi-hop con-

nection. In the following section, some theoretical issues are presented on how node mobility can be exploited to improve the capacity of an ad hoc network by contributing in the creation of communication opportunities. It is also highlighted that capacity can be traded for delay and a protocol is presented that allows this to happen.

## 3.1. Theoretical Foundation

The capacity of wireless ad hoc networks is strongly affected by phenomena like multi-path fading, path loss (see distance attenuation), shadowing by obstacles, and interference from other users. Therefore, it tightly depends on the total number of nodes of the network. Recent studies on the theoretical limit for this capacity have found that the throughput achievable per single source-destination pair decays approximately like $\frac{1}{\sqrt{n}}$ in the best case, and like $\frac{1}{\sqrt{n \log(n)}}$ in the worst case, $n$ being the total number of nodes[2] in the network [GK00]. These results hold true for *fixed* ad hoc networks with nodes located in random positions and supposed to be immobile.The throughput decrease, when increasing the number of nodes, is principally motivated by the presence of many concurrent transmissions over long distances between source and destination nodes that interfere with one another. As a consequence, the throughput is said to be *interference limited*. This is a negative result as it implies that ad hoc networks are not scalable. When nodes in the network are *mobile*, the previous results can improve by trading delay for capacity. Specifically, the throughput per single source-destination pair can be kept constant, even if the number of nodes increases [GT02] by reducing the interference effect of the concurrent communications as follows.

1. Sender nodes can transmit only if their respective receiver nodes are sufficiently close. In fact, transmissions over short distances need minimal power and cause less interference to the other source-destination pairs communicating in the same network. As a consequence, multiple transmissions between source-destination pairs can take place at the same time and the capacity can improve.

[2] Positions of nodes have been modelled with i.i.d. random variables.

2. Only some selected couples of source-destination nodes can transmit concurrently. These couples must be sufficiently distant each other to minimize the risk of interference during their communications. So, whenever two couples are too close, they are not allowed to communicate at the same time[3]. The throughput in this case is *distance limited*.

Allowing transmissions only in cases 1 and 2 is feasible only if any two nodes in the network (that can potentially become the source and destination of a message) are expected to get close to each other from time to time. This may only happen in case of *node mobility*. In addition, source nodes are required to have the possibility to *wait* to be within range of their respective destinations to transmit their packets, i.e., until a short-range communication opportunity arises, and should therefore *store* their packets till that moment. The routing policy they apply is known as *wait-for-destination*. It obviously increases the total transfer delay between the source and destination nodes and requires that the application which is responsible for the packets' exchange can tolerate some extra-delay (*delay-tolerant application*). To meet both the requirements 1 and 2 and increase the throughput achievable in the network, an alternative solution can be conceived by exploiting both node mobility and temporary storage of messages at intermediate nodes. This solution works as follows. Each source node *spreads* its packets towards its neighbour nodes by the means of local transmissions. The neighbour nodes act as relays and transmit the received packets to their respective destinations, when within reach, or alternatively to other relay nodes. Different relays receive different packets from the source node. So, a single copy exists in the network for each packet. Spreading packets all over the neighbourhood exploits *multi-user diversity* for forwarding because different packets flow through different relays, and consequently through different paths, towards the common destination node. Multi-user diversity therefore allows load balancing of the source traffic and also to reduce the end-to-end transmission delay because packets flow in parallel towards the destination. This method is successful when transmissions on the different paths do not interfere with one another. When multiple copies are

[3] Obviously not all the couple of nodes that are allowed to transmit, because they are sufficiently distant each other, actually decide to communicate. Only those of them that have some messages to transmit really transmit.

produced for each packet (adding some redundancy in the network) and then spread to different relays (and to different paths as a consequence) more guarantees are provided against uncertain channel conditions and network connectivity. Packet forwarding is said to make use of *path diversity* in this case. By exploiting multi-user diversity when nodes move around following i.i.d. random trajectories, it has been analytically proven [GT02] that relaying only once suffices like in the *2-hop relaying protocol* described below. Considering the time subdivided in slots, during the even time-slots packets are scheduled from sources either to relays or to destinations in the nearby, whereas during odd time-slots packets are scheduled from relays to destinations or again from sources to destinations if sufficiently close to each other. Specifically, during both the even and odd time-slots the scheduling policy selects, among all the possible node pairs those which are sufficiently close one another to allow direct communications take place. Node pairs that are selected can thus communicate simultaneously without interfering with one another. They can be, respectively, source-relay, source-destination, or relay-destination pairs. Then, communications may happen, assumed that the selected sources and relays actually have packets for those nodes they are allowed to transmit to. The above algorithm can be implemented both in a centralized and a distributed manner [SM04a]. In the latter case each node can decide, at each time and independently from the others, whether it wants to be a sender or a potential receiver. Source-to-relay and relay-to-destination transmissions may also occur concurrently during the same time-slot. In this case, even and odd time-slots should not be discriminated. Relay-to-destination and source-to-destination transmissions should always have absolute priority above source-to-relay transmissions. The 2-hop relaying protocol as described above provides no bounds to the delay that a packet can experience from source to destination. The delay increases with the number of nodes in the network. This has been proved both assuming the Random Way-point Mobility Model and the Brownian Mobility Model for the nodes in the network [SM04c] [SM04a] [SM04b]. Nevertheless, by adding some redundancy to transmissions in the 2-hop relaying protocol [SM04a], a good trade-off between delay and capacity is achievable. In this case, the source node transmits multiple copies of the same packet to different relays whenever a transmission opportunity arises. It eventually stops to transmit copies of the same

packet after having sent a certain maximum number of copies or after having received acknowledgement from the destination.

## 3.2. Experimental Studies

Many experimental studies have been conducted aimed at proving the possibility to exploit pair-wise contacts between mobile users in their everyday lives to support the routing system of an ad hoc network. The experiences, some of which are reported below, have focused on very different mobile communities from students in a university campus to citizens in their cities, to cars on highways or urban streets to wild species in their own ecosystems. All these experiences differ in many aspects:

**User mobility patterns:** users under study have been pedestrians, cars, whales, zebras, etc. They all move with different speeds and following either loose or constrained paths. Zebras move freely in the savanna, as do whales in the ocean. Students in a campus move relatively loosely even though in a smaller space. People and cars follow constrained paths, i.e., pavements and streets. Mobility is also affected by the particular situation that subjects are living and their intentions and scopes. So, a zebra is much slower when grazing than when escaping from a predator. Similarly, a car in a congested area is slower than when travelling along an empty street and a student when strolling in a yard is slower with respect to when running, on late, towards the classroom. Also the time affects mobility patterns, especially for human people. The density of people and cars in streets is different from day to night, as well as in different hours of the day (e.g., at rush hours with respect to other hours of the day).

**Communicating devices:** walking people generally carry laptops or PDAs; heavier and powerful instrumentation is instead suitable for car-to-car communications due to the absence of weight constraints; very light and waterproof sensors are applied on whales whereas zebras wear very light collars instrumented with sensors and solar arrays. All these devices are characterized by different transmission ranges.

**Interfering phenomena, disturbance:** while communications in savanna occur with practically no disturbance, in highways as well as in urban or suburban streets, communications are frequently subject to interference due to noise, intervening obstacles between the communicating peers, buildings and other solid surfaces that may cause reflections, etc. Underwater acoustic communications are very slow due to the means of communication itself, and eventually, in crowded places communications are difficult because of intervening people between the communicating peers, or because too many people wish to communicate at the same time and thus lead to network congestion.

**Connectivity pattern:** it results from the network density, the transmission range of the devices in use, and the interference present in the environment. In sparse networks good connectivity is achievable when the devices are powerful and have long transmission ranges. Dense networks on the other hand need lower transmission ranges but are more prone to communication errors due to congestion or interference. Definitely, it is quite difficult, or even impossible, to foresee the connectivity pattern of a real environment. From this comes the importance of the experiences described below that show what actually happens in the scenarios of references and also highlight key aspects to be careful to while designing network architectures and services. Unfortunately, as comprehensible, experimentation in real environments requires a great deal of effort and is very expensive as well. As a consequence, different experimental methods have been exploited and often real experiences have been partly mixed to simulations for the sake of simplicity as well as to avoid high costs.

### 3.2.1. College campus

This experience has recently been conducted in a university campus in Toronto, Canada. It demonstrates that it is possible to support delivery of messages among the participants of a community by simply exploiting pair-wise contacts among the participants themselves, in absence of any of the traditional ad hoc routing protocols. Messages are assumed to be forwarded with a random policy that reflecs

the randomness of the meetings between people. As people meet each other, they spread messages around, i.e., each message is sent to whoever is met and that has not already received the same message (epidemic forwarding). It has also been found that more sociable subjects are more successful relays for messages because they are interested in a higher number of pair-wise contacts and are thus able to deliver to the vast majority of the global community. Therefore, by exploiting the different forwarding attitudes of users it is possible to conceive more efficient forwarding policies than the epidemic forwarding policy.

Two distinct experiences [SCP+04] were conducted in different periods. During the first phase of both the experiences, some students (graduate students for the $1^{st}$ experiment and undergraduate students for the $2^{nd}$ experiment) were given Bluetooth-enabled PDAs to carry with them all over the day. PDAs kept track of the *pair-wise contacts* between students during their daily lives for some weeks. Some fixed PDAs were also hidden in places commonly frequented by students, so as to simulate the presence of base stations. They had no special feature implemented nevertheless. During the second phase of the experiences, the temporal sequence of the pair-wise contacts, as resulting from the data collected, were elaborated via simulation and the opportunities for multi-hop packet delivery were studied. Many opportunities were discovered as well as multiple redundant paths between couples of nodes. The experimentation proved that the user mobility can successfully be exploited to form ad hoc networks. In fact, nodes had significant contact and reachability in between one another. In addition, the connectivity patterns were not significantly affected by fixed nodes, meaning that the user mobility was mainly responsible for connectivity between nodes. It was thus concluded that potential networking gains would be possible if pair-wise communication were utilized, for example, to extend the reach of Wi-Fi hot-spots. The multi-hop reachability was studied by considering an *epidemic* forwarding policy. However, by analysing the successful delivery percentage of different next-hop neighbours it was found that some neighbours were more reliable than the others thus highlighting the possibility to conceive more efficient routing strategies. The results were quite encouraging, though obtained on a relatively small set of users (twenty in each experiment). By increasing the number of users involved, the network connectivity would surely improve. Another important result from this

experimentation is the importance of *power management* mechanisms in currently available mobile devices. During the experimentation a dramatic loss of data was experienced due to battery run out (devices were only equipped with volatile memory). Eight to ten hours of power supply were not enough considering that PDAs both monitored pair-wise contacts and also run appealing third-party applications (i.e., games), which could motivate users to carry the devices with them for as much time as possible. The experiments did not transfer any real data, detected connection quality, measured bandwidth, nor tracked user location.

### 3.2.2. Pocket Switched Networks

Pair-wise contacts can be characterized by the means of two parameters: *contact durations* and *inter-contact times*. The duration of a contact is the total time that two mobile nodes are within sight of each other and have thus the possibility to communicate. An inter-contact time is instead the time in between two contact opportunities. While the contact duration directly influences the capacity of opportunistic networks because it limits the amount of data that can be transferred between nodes, the inter-contact time affects the feasibility of opportunistic networks. In fact, they determine the frequency with which packets can be transferred between networked devices.

The University of Cambridge together with the Intel Research of Cambridge has recently studied [CHC+05] both the feasibility and capacity issues of opportunistic networks by analysing traces from experimentation similar to the one at the college campus described above. The first experiment to be considered was conducted at the University of California in San Diego (USA) and involved about $275$ PDA users, who were, specifically, students of the university campus [MV04]. The University campus had extensive $802.11b$ coverage in which students could roam with their PDAs. Coverage was provided by about $400$ Access Points (APs). The WiFi-capable PDAs periodically recorded information about all the APs that they could sense (the sampling period was about $20s$). The experimentation lasted for $11$ weeks. The second experiment [HKA04] was conducted at the Dartmouth College in Hanover (NH, USA), also in a college campus with overall $802.11b$ coverage provided by $566$ APs. During experimentation SNMP statistics were collected

at the APs over 17 weeks giving information (MAC and IP addresses) about the clients associated with each AP. The WiFi-capable devices which were monitored were both PDAs and laptops and also some VoIP phones. About 7000 users were monitored. Although the above experiments were not on mere ad hoc networks they contained precious information about user mobility in real environments. Hence, traces from these experiments were processed to yield contact information of a potential ad hoc network. Data was processed as follows. A contact opportunity was supposed to hold between two nodes every time they resulted to be simultaneously within sight of the same Access Point. This assumption is actually error-prone because even though in sight of the same Access Point, two devices might not be within reach of each other. On the other hand, two devices might be within range of each other but located in places where no instrumented Access Point is present, therefore their connection opportunities could not have the possibility to be logged. Despite these inaccuracies, the data collected can be considered good approximation of the actual data. In addition, it is a valuable source of data since it spans many months and include thousands of nodes. Further experimentation was conducted at the same University of Cambridge and Intel Research. Bluetooth-capable devices were used consisting of light weighted and highly portable iMotes[4]. They could be carried by people almost always all over the day. The same experimentation was repeated twice, at the Intel Research the first time and at the University of Cambridge the second time. The first experiment involved 17 researchers for 3 days while the second experiment involved 18 people, both PhD students and researchers, for 5 days. The iMotes performed periodical inquiries to discover and keep track of the other iMotes within sight or even of external Bluetooth-capable devices (the sampling period was 2-minute long). Hence, people involved were much more than those who had been enrolled initially and included whoever carried a Bluetooth-capable device. In fact, the choice of Bluetooth as the wireless technology for experimentation was mainly motivated by the vast diffusion of Bluetooth devices (headsets, PC sets, etc.) which could guarantee a thorough capture of the mobility patterns and pair-

[4] Made by Intel Research, the iMote platform derives from the Berkeley Mote (see http://www.xbow.com).The current version is based around the Zeevo $TC2001P$ system-on-a-chip that provides an ARM7 processor and Bluetooth support. A $950mAh$ CR2 battery provides power supply.

wise contacts. The analysis of the results led to an important result stating that both inter-contact times and contact durations are characterized by *heavy-tailed distribution functions* approximately following power laws. Moreover, power law functions have different coefficients depending on the different technology in use. This has interesting implications on the delay that each packet is expected to experience throughout the network. Specifically, it has analitically been proved that stateless forwarding algorithms cause an infinite expected delay, i.e., they do not converge, whereas algorithms that make use of multiple relays converge[5].

### 3.2.3. Vehicles moving on highways

Can vehicles on a highway form an ad hoc network and exchange messages in within? And further, can these vehicles be considered a community where occasional pair-wise contacts may help delivery of messages? Two studies are presented below which were conducted with the same methodology but had different scopes. They consisted of simulations on the ns-2 network simulator [WSc] with the mobility pattern of nodes generated by CORSIM (CORridor SIMulator) [OZRM00], which is a microscopic traffic simulator developed by the Federal Highway Administration at the US Department of Transportation. While the first study [BEH03] is interesting in that it analyses the characteristics of a Vehicle Ad hoc NETwork (VANET) in terms of topology changes, link durations, and redundancy of the paths, the second study [CKV01] is more on the opportunistic capabilities of an ad hoc network among the moving vehicles on a highway.

Results from [BEH03] highlighted that a VANET, on a highway, is subject to rapid network topology changes and to severe fragmentations. This is principally due to the relative speed of the vehicles running in both the same and opposite directions. In addition, the diameters of the connected network portions were very short. Simulations were run considering a highway section of about $15\ km$ with ten exits and ten on ramps and assuming that only $20\%$ of the total nodes were radio-capable. In addition, each node was considered to have a radio range of $150m$. The connectivity showed improvements as the transmission range also increased, however, each node could generally reach no more than $37\%$ of the other

[5] Specifics of the mentioned algorithms can be found further in this chapter.

nodes in the same highway section. Finally, link connectivity showed to suffer from rapid changes that often caused path disconnections before a complete data transfer could be concluded. As a result, utilization of routing algorithms relying on *route pre-calculation* was concluded to be *unsuitable* to such scenario. The simulation environment considered in [CKV01] consisted of a highway section of $10\ km$. Each node was supposed to have a fixed transmission range of $200m$ and to know its own position as well as those of the other nodes in the network. Message forwarding was performed on a hop-by-hop basis and followed a greedy strategy. Specifically, each message was forwarded towards that one of its next-hop neighbours which was *closest to the destination node*. Two variants of the routing strategy were investigated: the *optimistic* routing and the pessimistic routing According to the pessimistic policy, a packet was discarded each time no next-hop resulted closer to the destination with respect to the same forwarder node. According to the optimistic forwarding instead, in case no next-hop was available for forwarding, the packet was stored waiting for a subsequent forwarding opportunity. Results showed that optimistic forwarding contributed to increasing the delivery ratio even though at the price of a little longer delay (within $200s$). It was thus possible to conclude that a network that takes advantage of node mobility can operate with lower node density and maintain the same average delay. Results also showed that both the increasing number of lanes (better if greater than three) and the relative motion of vehicles in the two opposite directions contribute to the decrease of the end-to-end delay.

### 3.2.4. FleetNet

Vehicle Ad Hoc Networks (VANETs) were also investigated in the framework of the FleetNet Project which has been holding from $2000$ to $2003$ and has been partially founded by the German Ministry of Education and Research. The aim of the project has been the development of a communication platform to allow both vehicle-to-vehicle and vehicle-to-roadside communications. Vehicle-to-vehicle communications have the potential to enhance the drivers' safety and comfort, e.g., by giving them the possibility to receive emergency warnings from the vehicles ahead (flowing both in the same or the opposite lane), or allowing dis-

semination of traffic conditions that can be used to select alternative routes. In addition, vehicle-to-vehicle communications can allow the passengers of a vehicle to contact the passengers of the vehicles flowing around and, for example, play distributed games. Vehicle-to-roadside communications can allow access to the Internet and thus support commercial applications, for example, to inform the passengers of the vehicles of touristy points-of-interest or local services as well. The FleetNet solution has been tested via simulation and also, interestingly, by conducting some experiments in a real environment with real cars. The experimentation has proved that communications in VANETs are dramatically affected by buildings and any other kind of intervening obstacles in the traffic. Moreover, it has been found that realistic wireless transmission ranges are far shorter than those generally assumed in simulation studies to infer end-to-end paths. This discourages, once more, utilization of routing strategies relying on end-to-end precalculated paths in VANETs, because these can be considered neither reliable nor long lasting.

The platform deployed inside the FleetNet Project [WSa] [Enk03] maskes use of position-awareness of vehicles. Basically, vehicles are supposed to be equipped with GPS receivers to know their own positions and to exchange periodically their position information with their one-hop neighbours through *beacon messages*. Each vehicle is thus supposed to know the position of its immediate neighbours too. Message forwarding is managed greedily on a hop-by-hop basis. In the early stages of FleetNet, the *Greedy Perimeter Stateless Routing (GPSR)* strategy has been investigated. According to it, the next-hop node to choose for message forwarding should be the one-hop neighbour (with respect to the forwarder node) which is the nearest to the destination node of the message or, at least, the one whose direction is the nearest to the direction of the destination node. The introduction in the system of a *Reactive Location Service (RLS)* can then provide the geographical position of the destination node when it is not available. Nevertheless, further studies have found that the GPSR approach is not suitable to realistic city environments and is likely to fail. This is principally due to the physical constraints imposed by the real roads and streets to the mobility of vehicles. In fact, distances to the destination node should be measured on the real street directions and also forwarding directions should be mapped on actually available

route directions. The forwarding strategy has thus been substituted by the *Geographic Source Routing (GSR)*. It relies on city maps to select the best forwarding direction, i.e., the nearest direction, to that of the destination node, among those of the actual routes. Specifically, the sending node computes a path towards the destination node that can be traced on an underlying city map. The path is a sequence of junctions that the packet has to traverse in order to reach the destination. The sequence of junctions can either be put into the packet header or computed by each forwarding node to allow choosing always the most convenient next-junction. The efficacy of this solution has been tested on the ns-2 simulator [FFH+04] [LHT+03] [FMH+02]. To produce realistic results, mobility patterns of nodes have been accurately chosen to reflect the vehicle movements on city environments. Therefore, a well validated tool for the simulation of drivers' behaviours has been used. It is called FARSI and is used by the DaimlerChrysler AG to determine the lifetime of some parts of vehicles, such us shock absorbers or turn signals. During the simulations, transmission ranges have been supposed to be fixed ($500m$). In addition, source and destination nodes have always been chosen among those nodes that had an end-to-end connectivity holding in between. Results have shown that the GSR routing protocol outperforms both the DSR and AODV routing protocols (chosen as representative of topology-based routing protocols for ad hoc networks). Further simulation studies have been conducted by varying the wireless transmission range from $0$ to $4000m$ and also considering different communication patterns among the nodes observed. Assuming that every couple of nodes distant each other less than the communication range could communicate, it has been observed that a negligible number of partitions occur in the network when all the cars participate in message forwarding, both those flowing in the same and opposite directions. Network partitioning is less severe for transmission ranges greater than $500m$ [FKV02]. In addition to simulation studies, some practical experiences [MFHF04] have also been conducted with four $Smart^{TM}$ cars. They have been equipped with FleetNet routers that included IEEE $802.11b$ Wireless Network Interface Cards. Firstly, the performance limits of direct (single-hop) communications between pairs of static nodes have been tested. Results have shown compliance with the simulation studies. So have done the measures of the UDP and TCP throughputs resulting from multi-hop communications. Neverthe-

less, measurements obtained on-road with cars flowing in a real city environment, have demonstrated high sensitiveness to the intervening obstacles (e.g., encountered buildings and cars). Decreased throughputs and high packet loss have been experienced with respect to the simulation results demonstrating that the *environmental* factors deeply influence the radio propagation characteristics. Results have also shown that the end-to-end connectivity between the source and destination cars is almost always available either at one, two or three-hop distance. In absence of end-to-end connectivity, i.e., when no next-hop can be found available, packets have been discarded.

The difficulty of vehicle-to-vehicle communication in real environments has further been confirmed by other results obtained on-road [SBSC02] outside of the framework of the Fleetnet Project. By observing the characteristics of the wireless link holding between two vehicles in motion in different scenarios it has been found that a connectivity range can achieve up to $1000m$, but only under suitable, strict, driving conditions. The link quality measurements, for inter-vehicle separations up to $400m$, have demonstrated that a sub-urban environment is the most favorable and that urban driving conditions are the most hostile for inter-vehicle communications. The freeway-environment link quality lies in between the two. In addition, by varying the packet size as the average inter-vehicle separation varies in certain scenarios can help enhance the ad hoc network performance. The link quality or Signal-to-Noise Ratio (SNR) has been observed to degrade with the increasing distance. Throughput has also shown a decreasing trend with the increasing distance. Although the FleetNet routing approach does not actually make use of opportunistic paradigms and in fact it searches end-to-end paths between source and destination nodes, it can be considered opportunistic in that, at least in one version, packets are forwarded on a hop-by-hop basis without a priori knowledge of the entire source path (sequence of junctions) to the destination. In addition, as is shown from the real world experiences, transmission ranges are severly affected by realistic environment and generally reduced (about $150$ to $200m$ vs. $500m$ of the simulations). This naturally leads to an increased number of partitions in the global VANET. A similar environment could actually take andvantage opportunistic enhancements.

### 3.2.5. Ad Hoc City

An Ad Hoc City is a city where mobile nodes are allowed to communicate each other wirelessly or even access the Internet from anywhere because wireless coverage is granted all over the territory. Targeting realization of ad hoc cities, some researchers have recently proposed establishment in cities of backbone networks consisting of both fixed and mobile nodes. The fixed nodes should act as base stations sparsely deployed on the city territory, whereas mobile nodes should be implemented on fleets such as city buses, taxicabs, and delivery vehicles that naturally cover the city areas in both space and time. A mobile node wishing to communicate with another peer should simply wait or even look for an opportunity to send a message. The opportunity is represented by the mobile node to be within the communication range of a relay node. A bus, for example, could be nearing by chance or alternatively the sender node could intentionally reach the nearest bus stop where a bus will sooner or later arrive. Once transmitted to the first relay node (i.e., to the bus in this example), messages are forwarded to the destination node according to traditional ad hoc routing techniques (a reactive-based technique is used in this case) so as to limit the total propagation delay. In fact, the ad hoc city architecture is application-independent thus suitable for both delay-tolerant and delay-sensitive applications. Opportunistic techniques could be developed for this environment to support delay tolerant applications thus contributing to the decrease of the total network load and to eventually avoid congestion.

The Ad Hoc city is a multi-tier architecture [JHP+03] that supports wireless communications in a city environment. End-to-end connectivity between nodes is obtained by the means of a set of wired base stations spread in the city, and a mobile multi-hop wireless backbone consisting of city buses and delivery vehicles equipped with wireless devices. Nodes are classified in i) *personal mobile nodes*, such as PDAs and laptops, held by users moving around in the city, ii) *network mobile nodes*, which can be buses or delivery vehicles used as relays for message delivery, and finally iii) *fixed base stations* having the capability to access the Internet. Personal mobile nodes can be source or destination nodes of messages, whereas network mobile nodes and base stations can only be used for forwarding

purposes. Personal mobile nodes are not used for forwarding to save energy. A personal mobile node can access the Internet through a multi-hop path consisting of a sequence of network mobile nodes and a final base station. It can also send a message to another, far distant, personal mobile node over a path composed of a sequence of network mobile nodes till the closest base stations followed by a sequence of fixed hosts in the Internet till the base station closest to the destination node, and finally a sequence of network mobile nodes from the base station to the destination node. Two personal mobile nodes close each other can also communicate only by the means of network mobile nodes, i.e., without intermediate base stations. This architecture is novel in that it exploits node mobility to guarantee the end-to-end connectivity in a global city environment. Efficiency is granted from the network mobile nodes naturally covering the entire city area both in space and time, and thus creating a *backbone network* that can easily forward messages between any couple of nodes in the city. The *ad hoc city* also includes the proposal for a routing algorithm, *C-DSR (DSR enhanced for scalability)* and a simulation-based-on-traces study. The simulation has been conducted with the ns-2 network simulator. A realistic mobility pattern has been exploited which had previously been constructed on the basis of the actual traces of bus movements in Seattle, Washington, during their normal routes while providing passenger service throughout the city. The number of network mobile nodes has ranged between $750$ and $850$ and provided wireless service over an area of over $5000\ km^2$. Network mobile nodes have been assumed to have a radio range of $1.5\ km$. Packet-delivery ratio, packet overhead, path length and packet latency have been observed for communications between nodes randomly chosen among all the connected pairs of the network mobile nodes. The packet delivery ratio has ranged between $92$ and $97\%$, the median latency has been $42.52ms$ about, and the average overhead has been $222$ overhead packet transmissions per network node. The path has almost always been less then $6$ hops long, however, the majority of transmissions have been found to flow over two hops.

### 3.2.6. ZebraNet

While in the Ad Hoc City contact opportunities are exploited only to send messages to the mobile wireless backbone, i.e., for transmissions from the source node to the first relay node, the ZebraNet system is much more reliant on them. In fact, the two systems have a great difference in that, even though mobile, the base station in ZebraNet cannot cover the entire area of interest as the wireless backbone does in the Ad Hoc city. Therefore, all the nodes in the ZebraNet system are allowed to act as relays and exchange all their messages whenever they meet each other. By hopping node by node, messages are expected to eventually arrive to the base station when their carrying node gets within reach of the base station itself.

ZebraNet is an ongoing project [Sch02] [WSd04] at the Princeton University and a joint work of the Departments of Electrical Engineering, Computer Science, and Ecology and Evolutionary Biology. It focuses on information tracking of the wild species with the purpose to deeply investigate their behaviour and understand interactions and influences on each other. A thorough data gathering on wild species may also allow understand i) the migration patterns of wild animals, ii) how the human development into the wilderness areas affects the indigenous species, iii) how wild animals may be affected by changes in weather patterns or plant life, by the introduction of non-native species, or by other influences. The deployment scenario for the ZebraNet system is the vast savanna area of the central Kenya which is under control of the Mpala Research Centre [WSe]. So far, zebras have been the only species to be studied. The ZebraNet system consists of special collars that are carried by wild animals for data collection, a base station that periodically moves around in the savanna to gather information from the deployed collars, and a network protocol to percolate data from the collars towards the mobile base station. Collars include the necessary sensing equipment (e.g., a GPS receiver for position information) for periodic data sampling as well as some memory to store the collected data. They also include a processing unit to perform partial elaboration on the sensed data, and a short range and a long range radio for data transmission. The collar nodes form an ad hoc network and communicate among them in a peer-to-peer fashion. The base station is a mobile vehicle with researchers inside. Since both zebra nodes and base station are mobile and, in addition, the

base station is only sporadically present in the area under study, a complete data collection from all the zebra-nodes is very challenging. Two alternative protocols [JOW+02] have been considered for data collection, both exploiting contact opportunities among zebras. The first protocol is the *flooding protocol* and suggests that each collar sends its whole stored data to each one of the neighbours it discovers. As a result, each collar stores, together with its own data, the data received by its next-hop and multi-hop neighbours. Eventually, collars that can see the base station send the whole data collected to it. Therefore, when receiving data from a single collar, the base station collects the information sampled by many more collars. This methodology is very helpful to study a wildlife environment because there may be zebras which never approach the base station because generally live apart from the other herds, so, it is more likely that they encounter other zebras than the base station. The second protocol that has been tested is the *history-based protocol* and proposes that each node selects only one of its neighbours as relay for its data. The selected node is the one with the highest probability to eventually encounter the base station. Each node is assigned a hierarchy level (initially zero) that increases each time it encounters the base station, and conversely decreases after not having seen the base station for a certain amount of time. When sending data to a relay node, the neighbour to be selected is the one with the highest hierarchy level. To evaluate the performance of both the above protocols, some simulations have been run with a special simulator called ZNetSim. The simulator implements a realistic mobility model for zebras that relies on observations and statistics about the zebra behaviour conducted by biologists at the Mpala Research Centre. Simulation results have shown that both the protocols allow better *success rates* (i.e., the percentage of data that gets back to the base station) with respect to a *direct protocol* that only provides zebra-to-base station communications and not zebra-to-zebra communications. In addition, the history-based protocol performs better than the flooding protocol when considering the presence of storage and bandwidth constraints, particularly for radio ranges higher than $4\ km$ which lead to increasing the number of neighbours each zebra experiences. The history-based protocol also performs better from an energy saving standpoint because it limits the total number of exchanges between peers.

### 3.2.7. Networks on Whales

"Where there is a Whale there is a Way". This promise has been chosen to introduce a system for data gathering under the oceans. The objective of this project is similar to ZebraNet's, i.e., to collect data about wild species, specifically whales in this case, and investigate their habits as well as how they react to human intervention in their living environment. The system is composed by special tags applied to the whales for data monitoring and base stations that can be fixed (on buoys) or mobile (on seabirds). Communications are both whale-to-whale and whale-to-base station. Data is eventually forwarded on shore by base stations. This system is overall similar to ZebraNet, specifically the one version that makes use of a flooding protocol for data exchange, and shows another possible application scenario for opportunistic networks where the propagation delay of data is not a project issue. In addition, it should be noted that underwater transmissions generally rely on acoustic techniques that are characterized by non-negligible propagation delays. Therefore, applications conceived for this environment are mostly, necessarily, delay-tolerant and, as a consequence, naturally suited to opportunistic networking.

The Shared Wireless Infostation Model (SWIM) is conceived to collect biological data associated to whales in the oceans [SH03]. Data is monitored on whales by means of special tags that the whales themselves are equipped with. Tags can communicate with each other and exchange data. So, whenever two whales come close one another and their tags can communicate, a data exchange is performed. As a consequence, each whale stores its own data as well as the data collected by each whale it has met. The data is replicated and diffused through the whales while in motion and finally arrives to special SWIM stations located at some floating buoys on the ocean. Then, from the *SWIM stations* data is transferred to some place *on shore* for processing and utilization. SWIM stations have the potential for high speed communications (with respect to tag-to-tag communications), but are only sporadically connected to the whales because these are generally far away. No experimental results are available to demonstrate the efficacy of the SWIM system on real whales. However, realistic simulations have been conducted by carefully setting up environments and parameters with the results of observations

and studies conducted by biologists on whale habits. According to simulation results, the delay for arrival of data to the processing base stations is not negligible. However, it decreases as the number of the whales involved increases or even as the number or buoys increases. Better performance is also achievable when the buoys are well positioned, i.e., in places where whales are more likely to go. In fact, studies on the moving patterns of whales have shown that these mammals are such creatures of habit and periodically visit the same places, e.g., for feeding. Moreover, they tend to form groups (a group is generally formed by one female and different males), thus increasing the possibility to diffuse the data collected. Finally, when the SWIM stations are mobile, e.g., when they are put on seabirds, the system efficiency further improves.

## 4. Opportunistic Routing Techniques

Routing techniques in Intermittently Connected Networks (ICNs) typically aim to maximize the probability of message delivery. This probability is measured by the *delivery ratio* defined as the ratio of the total amount of data eventually arrived to destination to the total amount of data injected into the system. Another routing objective is minimizing the delay that each message experiences during delivery. It specifically consists of the time between when the message is injected by the source node and when it is completely received by the destination node. Given the frequent topology dynamics that characterize ICNs, both proactive and reactive routing approaches fail in finding appropriate routes for forwarding because they attempt to find complete paths between source and destination nodes, but these are likely not to exist. The most successful routing approach in ICNs is instead *per-hop routing* that strives to find a route on a hop-by-hop basis, i.e., by searching the most appropriate next-hop node only once having traversed the hop before. A next-hop node is chosen by exploiting local information about the contacts available at each hop towards other nodes as well as the queues of pending messages (waiting for forwarding) both at the forwarder node and at its neighbour nodes. This approach has the advantage to utilize information that is always fresh. Information can also derive from the underlying proactive or reactive rout-

ing layer. Proactive routing, for example, may be used to provide the set of nodes that are currently reachable and from which to select the preferred next hops. Reactive protocols instead, can be used to provide routes inside the connected parts of the network. Routing performance improves when more knowledge about the future topology of the network is available and exploited [SFP04]. Unfortunately, this kind of knowledge is not easily available and a trade-off must be met between performance achievement and knowledge requirement. The work in [SFP04] introduces four knowledge categories named oracles.

The **Contacts Summary Oracle** provides summary characteristics about contacts. It gives, for each couple of nodes, the average waiting time until the next contact is available. It thus provides time-invariant information.

The **Contact Oracle** gives information about contacts between any two nodes at any point in time. This is equivalent to knowing the time-varying multigraph of the ICN.

The **Queuing Oracle** gives information about instantaneous buffer occupancies (queuing) at any node, at any time. It can be used to route around congested nodes. Unlike the other oracles, the queuing oracle is affected by both the arrival of new messages in the system and the routing algorithm choices. It is probably the most difficult oracle to realize in a distributed system.

The **Traffic Demand Oracle** , in the end, answers any question regarding the present or future traffic demand. It is able to give information about the amount of messages injected into the system at any time.

Routing algorithms can be conceived to exploit one or more of the previous oracles. They assign costs to the edges of the network graph and compute a form of minimum-cost ("shortest") path. The cost of an edge is determined by consulting the available oracles and working out the delay that a message experiences in taking that edge. The delay components that are considered follow.

1. Queuing delay: the time until an edge (i.e., a contact) becomes available for transmisstion plus the time to drain the messages already scheduled for departure on that edge (contact).

2. Transmission delay: the time to inject a message completely into an edge (contact).

3. Propagation delay: the time to traverse the edge (contact).

The more oracles are available, the more realistic are the delays that are worked out, the more accurate are the routing decisions. Once assigned a cost to each edge, the shortest path can be computed using the Dijkstra shortest path algorithm. When no knowledge is assumed from the oracles, a message needing forwarding is simply delivered to the first contact available to the forwarder, or to a randomly chosen contact among those available. It is obviously the easiest protocol to implement but generally performs poorly with respect to more intelligent protocols. Solutions based on *message splitting* may be convenient in many cases. Messages are subdivided in multiple fragments so as single fragments can fit the single contacts available from time to time between two nodes. Different fragments may also be routed through different paths to reduce the delay or improve load balancing among multiple links (see Section 3.1). Obviously, effort must be spent to find multiple routes and appropriate fragment sizes. The work in [SFP04] proposes different routing algorithms, each of them using a different set of oracles. It demonstrates that the best solutions are those that make use of a greater knowledge about the future network topology and the incoming traffic. Unfortunately, the assumption of availability of the oracles is not actually realistic. The routing algorithms described below in this section do not rely on oraclesbut try some alternative solution. They can be categorized as follows. Further details on each single category are presented in the subsequent sections.

1. Dissemination-based routing;
    a) Replication-based routing;
    b) Coding-based routing;
2. Utility-based routing;
    a) History-based-routing;
    b) Shortest path-based routing;

   c) Context-based routing;
   d) Signal strength-based routing;
   e) Location-based routing;
3. Infrastructure-based routing;
4. Compulsory routing;
5. Carrier-based routing.

## 4.1. Dissemination-based Routing

Routing techniques based on data dissemination perform delivery of a message to destination by simply diffusing it all over the network. The heuristic behind this policy is that, since there is no knowledge of a possible path towards the destination node of a message, nor of an appropriate next-hop node, the message can only be sent everywhere. It will eventually reach the destination node by passing node by node. Dissemination-based techniques obviously work well only in dense or highly mobile networks where the contact opportunities needed for data diffusion are very common. They guarantee the shortest possible delay experience to messages, but they also are very resource hungry both in terms of memory occupancy and bandwidth usage and, as a result, lead to high energy consumption and poor scalability. Since it is not possible to obtain overall performance optimization, resource consumption is generally traded for delay. Memory of nodes is typically managed by techniques that include:

1. *Buffer Replacement techniques* to make room in the ingress buffer of a node when it is full and a new message arrives. These techniques avoid overwriting messages that have not been forwarded yet or that are not present in many copies in the network because their deletion could avoid final delivery to destination.
2. *Duplicate Detection techniques* to allow the destination node to recognize arrival of multiple copies of the same message so as it can drop them all.

3. *Garbage Collection techniques* to allow memory deallocation at intermediate nodes after arrival of the first copy of a message to destination.

Due to the considerable number of transmissions involved, dissemination-based techniques suffer from high contention and may potentially lead to network congestion. To increase network capacity what is generally done is to limit the spreading radius of a message by introducing a maximum number of relay hops allowed to each message or even limiting the total number of message copies present in the network at the same time. When no relaying is further allowed, each node can only be sent directly to destination when/in case met. Dissemination-based routing protocols may be subdivided in two categories: replication-based and coding-based. In replication-based protocols multiple copies of the same message are spread around in the network. In coding-based protocols instead, some processing is performed on the original messages before delivery. Blocks flowing over the network generally contain partial, coded information. The original messages can only be reconstructed at the destination node after having received a given number of coded blocks.

#### 4.1.1. Replication-based Routing

According to the **Epidemic Routing** protocol [VB00] messages diffuse in the network similarly to deseases or viruses, i.e., by means of pair-wise contacts between individuals/nodes. A node is *infected* by a message when it either generates that message or alternatively receives it from another node for forwarding. The infected node stores the message in a local buffer. A node is *susceptible* to infection when it has not received the message[6] yet but it can potentially receive it in case it comes into contact with an infected node (i.e., a node that stores that message). The infected node becomes *recovered* (healed from the desease) once having delivered the message to the destination node and, as a result, it also becomes *immune* to the same desease and does not provide relaying to the same message any more. More technically, epidemic routing works as follows. Each node possesses a buffer to store both the messages it generates and the incoming messages from other

[6] The message itself represents the infection/virus.

nodes that need forwarding. Messages are ordered in a list which is accessible through a hash table and is generally managed according to a FIFO policy. A summary vector describes which entries of the hash table correspond to actual messages and further contains a compact representation of them all. Each node has its own $ID$entifier ($ID$). Whenever two nodes come into communication range of each other (i.e., a pair-wise contact occurs) the one with the smaller $ID$, say $ID_1$, delivers its summary vector to the other node with the greater $ID$, say $ID_2$. Node $ID_2$ contrasts the received summary vector with its own summary vector and builds up a list with those messages stored by node $ID_1$ that it has not received yet (messages are discriminated by the means of unique identifiers and hop counts). Then node $ID_2$ makes request of the messages it misses to node $ID_1$ and afterwards waits for node $ID_1$ to send them to it. After messages have flown from node $ID_1$ to node $ID_2$, node $ID_2$ sends its proper summary vector to node $ID_1$ that repeats the same procedure as node $ID_1$. The choice of messages to ask for to the other node is based on considerations on the total buffer size available. A more intelligent choice may also keep into account the destination node of the announced messages. So, a node will preferentially request to the encountered node those messages that have as destination a node which is most frequently encountered. The procedure of message exchange is named *anti-entropy session* and is started whenever two nodes meet each other. Anyway, to avoid repeating this procedure uselessly, nodes are recommended to maintain a list of the most recently met nodes and to start exchanges only with nodes that do not belong to this list. Dissemination throughout the network guarantees that messages eventually arrive to their actual destinations. The dissemination process is somehow bounded because each message when generated is assigned a *hop count limit* giving the maximum number of hops that that message is allowed to traverse till the destination. When the hop count limit is set to one, the message can only be sent directly to the destination node. The greater is the hop count limit of a message the more largely it is spread around in the network and, as a result, the more chances it has to eventually reach the destination. Moreover, the delivery latency is more likely to be low. When the hop count limit is imposed to be low, a message cannot be frequently forwarded and mostly moves from point to point thank to the mobility of the nodes that store it (delivery exploits mobility more than wire-

less transmissions). Simulation studies have shown that epidemic routing allows delivery of messages in disconnected ad hoc networks where traditional routing algorithms generally fail. By providing vast and rapid diffusion of messages all over the network, epidemic routing maximizes the delivery success rate of messages and minimizes the latency experienced by each message. However, this protocol tends to consume great deal of resources, specifically *storage capacity*, network bandwidth for transmissions, and energy for both message storing and sending. High storage capacity and hop count limits lead to both high delivery success rate and low latency[7]. On the other hand, the total resource requirement of the system also increases. $100\%$ delivery success rate would be granted if each node could store all the messages flowing in the network at a time. Anyway, it has been demonstrated that good delivery success rate and reasonable latency are possible if only can each node store $5 - 25\%$ of all messages generated in the network at a time. Alternatively, it would also be good if some nodes had considerable storage capacity to act as backbone nodes whereas other nodes only had minimal buffer capacity.

Clearly, the efficacy of epidemic routing, as originally conceived, is severly limited by the scarcity of the storage capacity available at nodes. After generation of a message, many copies of it are spread all over the network and continue to be stored even after arrival of one of the replicas to the destination. This obviously leads to memory wastage. A message is only deleted when a new message arrives to a node needing storage and the buffer is full. Hence, to make room for the new message, an old message is selected from the buffer to be dropped and finally replaced with the new one. More effective techniques for buffer replacement are investigated in [DFL01]. Specifically, four alternative strategies are considered:

*Drop-Random (DRA)* : a packet is dropped at random. This policy gives delivery priority to the messages that are generated by the most silent sources (i.e., sources with a low packet generation rate). In fact, the more are the packets generated by the same source, the more are the packets from that source that might be stored in the same buffer, and the more likely is for one of them to be dropped.

[7] This is because many nodes can contemporary store the message thus increasing the chances that one of them soon meets the final destination.

*Drop-Least-Recently-received (DLR)* : messages that are dropped first are those that have spent much more time in the buffer. In fact, these messages are more likely to have already reached destination.

*Drop-Oldest (DOA)* : messages that are dropped first are those that have spent much more time in the network. These messages have the highest probability to have already been delivered to destination.

*Drop-Least-Encountered (DLE)* : the message to be dropped from the buffer is the one with the worst estimated *likelihood of delivery*. The likelihood of delivery of a message corresponds to the likelihood of the node that holds that message to be within reach of the destination node of that message or alternatively to be within reach of another node likely to encounter the destination. Each node is supposed to maintain a list of likelihoods of delivery, one for each known node in the network. This list is updated after pair-wise contacts. Let $M_t(A,\ B)$ be the likelihood of node $A$ to be within reach of node $B$ at time $t$. If node $A$ encounters node $B$, it increases the likelihood of delivery to node $B$ which has just been met, as well as the likelihood of delivery to other nodes that $B$ is likely to encounter (i.e., if node $B$ is likely to encounter node $C$ and node $A$ is likely to encounter node $B$ then node $A$ is likely to successfully deliver to node $C$ through node $B$). Obviously, node $B$ appropriately updates its list of likelihoods of delivery too. Eventually, likelihoods of delivery are periodically decreased in absence of pair-wise contacts. The following formula expresses analytically the heuristic above in case node $A$ meets node $B$. $\alpha = 0.1$ and $\lambda = 0.95$ respectively.

$$M_{t+1}(A,\ C) = \begin{cases} \lambda M_t(A,\ C) + 1 & if\ C = B \\ \lambda M_t(A,\ C) + \alpha M_t(B,\ C) & for\ all\ C \neq B \\ \lambda M_t(A,\ C) & if\ none\ met \end{cases} \quad \text{(IV.1)}$$

Simulation results show that the *Drop-Oldest* and the *Drop-Least-Encountered* strategies are those that perform better with respect to the other strategies. Moreover, for fixed buffer sizes at nodes and increasing network load, the *Drop-Least-Encountered* algorithm outperforms the *Drop-Oldest* algorithm.

The **MV routing protocol** [BBL05] is a further step beyond epidemic routing. Messages are exchanged during pair-wise contacts through the same *anti-entropy session* introduced in [VB00] and [DFL01]. However, the MV protocol introduces a more sophisticated method to select the messages to request to a node which is encountered. Basically, the choice depends on how much confident a node is to provide to these messages successful delivery towards their respective destinations. Hence, each node computes the delivery probability towards each other node in the network (each node is a potential destination node). The delivery probability relies on observations on both the *meetings* between nodes (like in [DFL01]) and the *visits* of nodes to geographical locations which have occurred in the recent past. The name *MV* protocol itself comes just from *Meetings* and *Visits*. The delivery probabilities are worked out as follows. Let $P_0^k(i)$ be the probability that node $k$ visits region $i$. This also corresponds to the probability to deliver a message successfully and without any intermediate relay to any node that is located in the same region $i$. It may be computed according to the following formula.

$$P_0^k(i) = \frac{t_i^k}{t} \tag{IV.2}$$

Where $t_i^k$ is the number of time units in which node $k$ has visited region $i$ during the last $t$ time units (a time unit is an arbitrary period of time, i.e., any period of reference). Let $P_1^k(i)$ be the probability of node $k$ to deliver a message to a node located in region $i$ through exactly one intermediate node. This probability corresponds to the probability for node $k$ to meet an intermediate node, i.e., any $j$ node, which subsequently visits region $i$. It can be obtained as follows.

$$P_1^k(i) = 1 - \prod_{j=1}^{N} (1 - m_{jk} P_0^j(i)) \tag{IV.3}$$

Where $N$ is the total number of nodes and $m_{jk}$ is the *meeting probability* that nodes $j$ and $k$ visit the same region simultaneously and thus have a contact opportunity. $m_{jk}$ is computed as follows.

$$m_{jk} = \frac{t_{j,k}}{t} \tag{IV.4}$$

Where $t_{j,k}$ is the number of time units that nodes $j$ and $k$ have visited the same region simultaneously during the last $t$ time units. The probability that a message from node $k$ to node $j$ traverses exactly $n$ relays is instead given by:

$$P_n^k(i) = 1 - \prod_{j=1}^{N} (1 - m_{jk} P_{n-1}^j(i)) \tag{IV.5}$$

Together with the observation of the motion pattern of nodes in the network, the MV protocol also takes into consideration the possibility to change this motion pattern to facilitate forwarding operations and increase the network capacity. Therefore, special nodes called *Autonomous Agents* are added to the network. They are mobile robots, both ground-based and airborne and move around in the network area following pre-computed routes which are appropriately designed to optimize contribution to data collection and forwarding and to network performance improvement in general. Unfortunately, computation of optimal paths is notoriously an NP-complete problem. In MV routing sub-optimal motion patterns are found by the means of a *multi-objective controller system* which is taken from the robotic control theory. A *controller* is the algorithm used to generate the motion of an agent. In a *multi-objective controller system* different controllers are defined with different scopes:

1. *Total Bandwidth Controller*: it aims to reach first those nodes which store the greatest number of undelivered messages;
2. *Unique Bandwidth Controller*: it aims to visit first those nodes that are the only repositories for their messages;
3. *Delivery Latency Controller*: it strives to visit first nodes that are characterized by the highest delivery latency;
4. *Peer Latency Controller*: starts visiting the least recently visited nodes.

These controllers' goals cannot be reached all together at the same time hence, it is necessary to find some reasonably appropriate combination. Two compositions have been investigated in [BBL05], namely: *Nullspace Composition* and *Subsumption*, both from the multi-objective control theory. The MV protocol has

been tested via ns-2 simulations assuming that transmissions occur from mobile to stationary nodes. According to the mobility model that used in simulations, nodes generally move towards three points of attractions: a home location and two different remote locations. Nodes are attracted from the home location $50\%$ of time, from the other location $25\%$ of time each. Simulation results show that the MV protocol outperforms FIFO queuing-based protocols and that the addition of autonomous agents significantly increases the delivery rate of the network. Moreover, MV routing also allows lowering of the message latency. With regard to the control strategy, the Nullspace Composition outperforms the Subsumption approach. The protocol well reacts to the *offered load* increase.

The **Spray and Wait** protocol [SPR05] is inspired to flooding-based delivery schemes in that it makes no use of information on network topology or knowledge of the past encounters of nodes, however, it significantly reduces the transmission overhead by limiting the total number of copies that can be transmitted per single message. It is thus more energy efficient than flooding-based protocols. Moreover, the delay experience of messages is quite similar to the optimal case of epidemic routing. Finally, the Spray and Wait protocol is interestingly robust and scalable. It works as follows. Message delivery is subdivided in two phases: the *spray phase* and the *wait phase*. During the spray phase multiple copies of the same message are spread over the network both by the source node and those nodes that have first received the message from the source node itself. This phase ends when a given number of copies, say $L$, have been disseminated in the network. Then, in the wait phase each node holding a copy of the message (i.e., each relay node) does nothing but simply store its copy and eventually deliver it to the destination when/in case it comes within reach. The spray phase may be performed in many ways. According to the *Source Spray and Wait* heuristic, the source node forwards all the $L$ copies of the message to the first $L$ distinct nodes it encounters. On the other hand, according to the *Binary Spray and Wait* heuristic, the source node of the message hands over the first $\lfloor \frac{L}{2} \rfloor$ copies of the message to the first node encountered and holds the remaining $\lceil \frac{L}{2} \rceil$ copies. Then, any node $A$ storing $n > 1$ message copies (either the source or a subsequent relay) and being within reach of another node $B$ that possesses no copies of the message, hands over $\lfloor \frac{n}{2} \rfloor$ copies to $B$ and keeps the remaining $\lceil \frac{n}{2} \rceil$ copies for itself. When a node is left with

only one copy of the message, it switches to *direct transmission* and only transmits the message to the final destination node when/if met. Under the assumption that nodes move around according to an i.i.d distribution, the Binary Spray and Wait routing outperforms all the other spray and wait routing algorithms allowing the minimum expected delay. An interesting characteristic of Spray and Wait routing is that it can be appropriately tuned to achieve the desired performance in a specific scenario. Specifically, the number of copies of a message, L, can be set to achieve a given expected delay. Details on the tuning process of L can be found in [SPR05]. Spray and Wait is extremely scalable, in fact, as the number of nodes in the network increases, the percentage of nodes that need to become relays to achieve the same performance actually decreases, whereas most of the other multi-copy schemes perform a rapidly increasing number of transmissions as the node density increases. Performance of Spray and Wait has been tested theoretically and via simulation. Simulations have been conducted on a custom discrete event-driven simulator with nodes moving according to the random waypoint mobility model. They show that Spray and Wait clearly outperforms other multi-copy (e.g., epidemic routing) or even single-copy routing protocols (e.g., see below utility-based protocols) in terms of both number of transmissions and delay experience of messages. Among the replication-based routing protocols it may also be included the *2-hop relaying protocol* already described in Section 3.1.

#### 4.1.2. Coding-based Routing

In **Erasure-Coding Routing** [WJMF05] messages are encoded at their source node and for each message many smaller *code blocks* are generated. Specifically, if $M$ is the length in bytes of the message, the number of code blocks that is generated equals $\frac{Mr}{b}$ where $b$ is the length of a single code block ($b < M$) and $r$ is the constant *replication factor*. After generation, code blocks are equally distributed to $kr$ relays, $k$ being a constant value. Then, the algorithm needs $\frac{(1+\varepsilon)M}{b}$ code blocks to arrive to destination to reconstuct the original message: any $\frac{(1+\varepsilon)M}{b}$ over the total $\frac{Mr}{b}$ blocks. $\varepsilon$ is a constant that depends on the particular encoding algorithm which is used however, it can be considered sufficiently small to be neglected, and it can be stated that only $\frac{1}{r}$ of the total code blocks are needed to arrive to

destination to decode the original message. This helps limit the total propagation delay from the time the message's blocks are encoded and sent by the source node till the time the destination decodes the message itself. In fact, given the high number of relays involved, there is a big chance that at least $\frac{1}{r}$ of them, i.e., as many as required, are reliable and fast. For the same reason Erasure-Coding is quite robust against packet loss due to bad channel quality or congestion. Moreover, it does not suffer from increased propagation delay due to bad choice of forwarding relays or paths. It introduces controlled protocol overhead in terms of bytes transmitted and is very energy-efficient. Erasure-Coding routing has been tested analytically and also via simulation with `dtnsim`, which is a discrete event simulator for DTN environments. Two mobility patterns have been tested: the former from the ZebraNet system [Sch02] [WSd04] [WSe] [JOW+02] and the latter based on heavy-tailed inter-contact times (see considerations in [CHC+05]). Results show that Erasure-Coding Routing well scales with node density and network size. It has the best worst-case delay with respect to both flooding-based and history-based techniques.

In **Network Coding Routing** [WLB05] packets travelling over the network are obtained by combining the packets that are actually generated at the source nodes. Initially, source nodes broadcast the packets they generate. Intermediate nodes receive packets from both source nodes and other intermediate nodes, then they perform a combination of the whole information received and ultimately send out the resulting combination packets. Packets are disseminated all over the network and eventually arrive to destination where the original packets are reconstructed by running the decoding process. Let $A$, $B$, and $C$, be the only three nodes of a small network. Let node $A$ generate the information $a$ and node $C$ generate the information $c$. Then suppose the information produced needs to be known at all the nodes. Hence, node $A$ and node $C$ send their information to node $B$, then node $B$ rather than sending two different packets for $a$ and $c$ respectively, broadcasts a single packet containing $a \oplus c$ ($\oplus$ staying for $XOR$). Once received $a \oplus c$, both nodes $A$ and $C$ can finally infer the missing information i.e., node $A$ can infer $c$ and node $C$ can infer $a$ (see also Fig. IV.1). According to the protocol described in [WLB05], given any node $v \in V$ in the network ($V$ gives the complete set of nodes in the network) and $N(v)$ the set of its neighbours reachable through

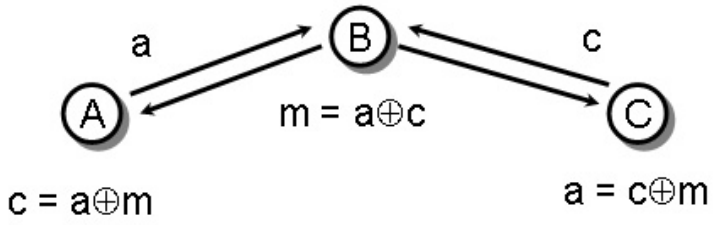


Figure IV.1.: Example of network-coding efficiency.

physical layer broadcast, whenever a set of *information vectors* $x_i \in \{0,\ 1\}^k$ (for some $k$, $i = 1,\ \ldots,\ m$) is received at node $v$, a linear combination is worked out as follows and sent to the neighbours.

$$\sum_{i=1}^{m} g_i x_i \tag{IV.6}$$

Where $g_i$ is the *encoding vector* for $x_i$. Information vectors are always sent together with the corresponding encoding vectors. Each receiving node stores the incoming tuples $< g_i,\ x_i >$ together with the tuples corresponding to the packets generated on its own of the form $< e_i,\ x_i >$ where $e_i$ is the *i*-th unit vector. All the tuples are collected in a decoding matrix $G_v$ whose rank gives the number of packets that can be decoded at the moment. If $m$ is the total number of packets generated in the network, each node can decode all of them when its decoding matrix will have rank $m$. When a new packet arrives to a node, it is said to be $innovative$ if it increments the rank of the decoding matrix, otherwise it is simply ignored. Given the forwarding factor $d > 0$, when an innovative packet is received, $\lfloor d \rfloor$ vectors will be generated from the corresponding matrix and broadcast to neighbours. A further vector is generated and sent with probability $d - \lfloor d \rfloor$. In case of a source node, $\max\{1,\ \lfloor d \rfloor\}$ vectors are generated and broadcast. Moreover, an additional packet is sent out with probability $d - \lfloor d \rfloor$ if $d > 1$. In other words, a new packet is sent out by the source at least once. To avoid managing matrices too extended, when many sources are present in the network, each producing different packets, source vectors are grouped into *generations*. There exists one matrix per single generation and only vectors of the same generation can be combined. The work in

[WLB05] proposes hash functions to be used to distinguish generation members. $\Gamma_\gamma = \{x_{i,\,j} \mid f(x_{i,\,j}) = \gamma\}$ is the set of all the source vectors of the generation $\gamma$. Where $x_{i,\,j}$ is the *j*-th information vector that originates at the *i*-th source. $f(x_{i,\,j})$ determines which generation the packet belongs to. A new hash function is generated whenever the matrix becomes too big. Once a node has decoded the whole generation, the corresponding matrix is no longer of use to the node. However, its information might be required by neighbours in order to be able to decode. A node should therefore keep info from a matrix, as long as the node is still eligible to generate packets from it. To save memory, a node may reduce the rank of the matrix over time. Network coding routing has good performance in terms of packet delivery ratio both in dense mobile networks and sparse networks with high packet drop rate and nodes that sleep for a considerable amount of time. It takes great advantage from node mobility with respect to other stateless routing protocols.

## 4.2. Utility-based Routing

Dissemination-based routing allows optimal message delivery rate and also minimizes the delay experienced by each message during delivery. Unfortunately, it consumes plenty of bandwidth and storage capacity and is thus very energy inefficient. The protocols presented in this section deal with delivery of messages in intermittently connected ad hoc networks while striving to optimize the overall bandwidth usage and the energy consumption of nodes. As common strategy, they limit the number of copies of a message that are spread over the network and perform an appropriate selection of the relay nodes. Each message is forwarded only to one or few relays, those which have the highest probability to deliver successfully the message towards the destination node. This avoids unnecessary message replicas, exchanges, and also retransmissions. The usefulness of a host as next hop for a message is hereafter referred to as *utility* of that host. Albeit optimizing the bandwidth usage and the overall energy consumption of the network, utility-based techniques slightly increase the delay that each message experiences during delivery with respect to the delay resulting from dissemination-based techniques. This is due to possible errors and inaccuraces in selecting the best relays. Moreover,

utility-based techniques have higher computational costs than replication-based techniques. Nodes need to maintain a state that keeps track of the utility values associated to all the other nodes in the network (i.e., all the possible destination nodes), and hence need storage capacity for both state and messages. Finally, the cost to hold and constantly update the state at each node should be further considered in the overall protocol overhead. Utility-based routing protocols can be further subdivided in:

1. *History-based routing protocols*: these protocols work out utilities on the basis of the history of encounters between nodes or even the history of visits to locations. As representative for this category the following protocols are presented: a) the Disconnected Transitive Communication (DTC) protocol, b) the Probabilistic Routing Protocol using History of Encounters and Transitivity (PROPHET), and c) the MobySpace routing protocol.

2. *Shortest path-based routing protocols*: they aim to minimize the total delivery cost of messages and thus strive to select the minimum cost path. It generally corresponds to the path that can be traversed with minimum delay. The protocols presented here for this category are: a) the Extremely Opportunistic Routing protocol (Ex-OR), b) the Shortest Expected Path Routing protocol (SEPR), and c) the Practical Routing protocol.

3. *Context-based routing protocols*: they work out nodes' utilities to forward messages by doing more general considerations than the previous protocols. They consider, for example, nodes' mobility and residual battery levels. Examples of context-based routing protocols are: a) the Context-Aware Routing protocol (CAR) and b) the Interrogation-Based Routing pRotocol (IBRR).

4. *Signal strength-based routing protocols*: the next hop is chosen to be the one connected to outgoing link with the highest signal strength. An example is briefly introduced which is named Signal strength assessment Based Routing enhancementS-OLSR (SBRS-OLSR).

5. *Location-based routing protocols*: they typically strive to locate the destination node of a message firstly by the means of an underlying classical routing protocol, either proactive or reactive. In case the destination cannot be

found some attempt is made like, for example, performing local broadcast and trying to find a connected path to the destination from the neighbours reached, or alternatively nearing the last known location of the destination and spraying the message in the nearby. Examples of location-based routing protocols are: a) the Spraying Protocol and b) the Mobile Relay-based Protocol (MRP).

### 4.2.1. History-based Routing

The **Disconnected Transitive Communication (DTC) model** [CM01] refers to an overall communication model for intermittently connected networks. It includes a traditional network-layer routing protocol (e.g., DSR) to manage communications inside the connected clusters of the network and a novel middleware-layer routing protocol for cross-cluster communications. The middleware-layer routing protocol therefore intervenes only when the traditional network-layer routing fails to transmit a message to the destination host and selects the best next-hop node possible. The best next-hop node for forwarding is chosen to be that host which is inside the cluster and likely to come within reach of the destination node earlier than the other nodes in the cluster and than the current message holder itself. It can thus be considered the one host closest to the destination node. The utility of a node to act as forwarder towards a certain destination node is worked out by exploiting the history of encounters for that node (i.e., the past connectivity pattern), its future expected mobility pattern (that should be explicitly entered), and its residual energy level. To obtain the history of encounters the node overhears[8] the traffic generated by the other nodes' and captures the identity of all the nodes in the nearby. Then two lists are arranged, the first with the *most recently noticed* nodes, and the latter with the *most frequently noticed* nodes. Each entry, of both the lists, contains the identity of a node encountered (*noticed*) and the time of the most recent contact. The utility components are worked out from the two lists as

[8] The Hello packets originated at the one-hop neighbours are overheard. The Hello packets are added an extra-bit that gives information about the dynamicity of the nodes that are farther than the one-hop neighbours.

follows.

$$U_{MRN} = \left(1 - \frac{CurrentTime - TimeLastNoticed_D}{TimeOut_m}\right) \cdot 100 \qquad \text{(IV.7)}$$

$$U_{MFN} = \left(1 - \frac{CurrentTime - FirstTimeNoticed_D}{NumTimesNoticed_D \cdot TimeOut_m}\right) \cdot 100 \qquad \text{(IV.8)}$$

$U_{MRN}$ is the utility component obtained from the Most Recently Noticed list whereas $U_{MFN}$ is the utility component obtained from the Most Frequently Noticed list. $TimeOut_m$ is the residual time-to-live of the message and is used to normalize both the utility functions. $CurrentTime$ is the current wall-clock time. $TimeLastNoticed_D$ is the wall-clock time of the most recent contact with the destination node $D$. $FirstTimeNoticed_D$ is the wall-clock time of the oldest contact with the destination node $D$, whereas $NumTimesNoticed_D$ is the number of contacts with the destination node $D$ since the $FirstTimeNoticed_D$. The heuristic behind these formulas is that the utility of a node as forwarder towards node $D$ is as much higher as more recent has been the last meeting with node $D$ as well as more frequent have been the meetings with node $D$. If the destination host has never been encountered or the right hand side of the equations are negative then $U_{MRN} = 0$, $U_{MFN} = 0$. In case future meetings with the destination node are scheduled (e.g., on the calendar of a PDA) or even they are predictable, it is possible to compute the following utility component ($U_{CAL}$ comes from calendar).

$$U_{CAL} = \left(1 - \frac{NextMeetingTime_D - CurrentTime}{TimeOut_m}\right) \cdot 100 \qquad \text{(IV.9)}$$

$NextMeetingTime_D$ is the wall-clock time of the next meeting with node $D$. The utility component $U_{CAL}$ is as much lower as farther is the time of the next meeting with node $D$. If no encounters are scheduled then $U_{CAL} = 0$. Another utility component is defined by considering the residual energy level of the node. Specifically, the longer the host will remain alive, the higher the probability that it will meet the destination:

$$U_{power} = \left(1 - \frac{TimeOut_m}{EstimatedRemainingPower}\right) \cdot 100 \qquad \text{(IV.10)}$$

$EstimatedRemainingPower$ is the residual battery level of the node. The utility components are summed up, with different weights, into a unique utility. The result is much more effective as much data is used for computation (e.g., the greater the number of the most recently encountered nodes). However, since each node has to maintain and store a history of encounters for each potential destination node, a great storage capacity would be needed if no limitation was introduced to the length of the lists. A trade-off must thus be met between the effectiveness of the utility estimate and the storage capacity required to hold the history of encounters. To generate and update the lists, a *discovery protocol* is run periodically. It includes three phases: i) *utility probe phase*, ii) *utility collection phase*, and iii) *redistribution phase*. During the utility probe phase, the host that has a message to send to a certain destination host probes its current cluster to collect the utilities that each current member of the cluster has associated to the destination host of the message to send. The message sent for probe includes the destination host of the message to send, the set of weights to associate to the utility components, the time-to-live value of the message, and a threshold value. The nodes which are queried calculate their utilities for the specified destination host by utilizing the weights announced in the probimg message. Then each node transmits the utility calculated to the probing node only in case its value is greater than the threshold indicated in the probing message. The threshold value contributes decreasing the overall network traffic injected into the cluster. The time-to-live value of the message is used for garbage collection. When it expires the message is dropped from the system. After the probing phase, the utilities are collected at the probing node and a decision is made about the best host (utility collection phase). Finally, in the redistribution phase, the message is sent to one or more nodes in the cluster: that/those selected as best next hop/(s). The *re-discovery interval (RDI)* is tunable by the application and is able to approximately detect changes in the cluster membership without adding a great deal of network overhead. The interval is shortened when cluster membership changes are more frequent and lengthened when changes are infrequent. Another utility component can be defined to depend on the re-discovery interval. Higher utilities are assigned to the most active

hosts, i.e., those with lowest RDIs.

$$U_{RDI} = \left( \frac{Min_{RDI}}{RDI} \right) \cdot 100 \qquad \text{(IV.11)}$$

$Min_{RDI}$ is the minimum rediscovery interval for a host and is based on the environmental characteristics of the place where the DCT communication protocol is being employed. When the RDI is small, a high load is placed over the network to transmit the utility probes and the response packets. If too large the hosts run the risk of not taking advantage of transient connections with hosts with very high utilities. A possible trade-off consists in doubling the RDI each time a utility probe is completed. This causes slowing down exponentially the frequency of rediscovery. Nevertheless, when a new host is detected in the cluster then the RDI must be immediately reset to the initial value, so a quick rediscovery is forced. A more flexible solution for utility calculation can exploit *code mobility*. The source node of a message is in charge of building up the most appropriate algorithm for utility calculation according to the application requirements. The algorithm may make use of attributes and parameters that are host-specific as well as give more priority to some specific hosts with respect to others. When a node is probed, it runs the source-defined utility function (which is to be distributed together with the probing messages) and thus produces the utility to return to the source. Obviously this approach raises tricky security issues related to code mobility. Some further protocol optimizations are possible:

1. Some degree or replication could be introduced so a message could be sent to more than one next hop at a time. However, bandwidth usage would increase too.

2. More effective routing decisions could be possible if network layer information were exploited but the dependence on the routing protocol would increase.

3. Probing messages are generally broadcast. This gives the protocol considerable overhead, thus techniques to reduce this load should be introduced.

4. Another factor that increases the total burden of the system is the calculation

of unicast routes from the queried nodes to the source node. This should be alleviated too.

Similarly to the above Disconnected Transitive Communication model, in the **Probabilistic Routing Protocol using History of Encounters and Transitivity (PROPHET)** [LDS03], each node holds a delivery predictability table with the probabilities of successful delivery towards each known node in the network. Also in this case, the probability to deliver a message to a certain destination node increases whenever it comes within sight, and decreases over time in case no meeting occurs. The *Delivery Predictability* from any node $a$, towards a destination node $b$ is indicated as $P_{(a,\ b)} \in [0,\ 1]$. Every time two nodes $a$ and $b$ meet each other, their delivery predictabilities increase as follows.

$$P_{(a,\ b)} = P_{(a,\ b)_{old}} + (1 - P_{(a,\ b)_{old}}) \times P_{init} \tag{IV.12}$$

$P_{(a,\ b)_{old}}$ is the last known delivery-predictability value, whereas $P_{init} \in [0,\ 1]$ is an initialization constant. The delivery predictabilities decrease over time, as indicated below, when the corresponding nodes do not meet each other.

$$P_{(a,\ b)} = P_{(a,\ b)_{old}} \times \gamma^k \tag{IV.13}$$

$\gamma \in [0,\ 1)$ is the *aging constant* and $k$ is the number of time units that have elapsed since the last time that delivery predictability has been aged. Delivery predictabilities are also updated using the *transitive property*. Basically, if a generic node, say $a$, is likely to encounter another node, say $b$, and this second node $b$ is likely to encounter another node, say $c$, it can be inferred that the first node $a$ is a good relay node for the third node $c$ and vice-versa.

$$P_{(a,\ c)} = P_{(a,\ c)_{old}} + (1 - P_{(a,\ c)_{old}}) \times P_{(a,\ b)} \times P_{(b,\ c)} \times \beta \tag{IV.14}$$

$\beta \in [0,\ 1]$ is a scaling factor and gives the impact of transitivity on the delivery predictability. Every time two nodes are within sight of each other, they exchange the summary vectors of the messages they store. They add to each message entry their delivery predictability related to the destination node of that message (i.e., how likely is that it can deliver successfully that message it is holding). Both

nodes, then, decide which messages to request to the other, based on these probabilities. Specifically, node $a$ requests a message from node $b$ which is destined to node $c$ only if the delivery predictability from node $a$ to node $c$ is higher than the delivery predictability from node $b$ to node $c$. The PROPHET algorithm has been tested via simulation (the simulator had been made from scratch) for networks with nodes moving according to the *random waypoint* mobility model firstly and to a more realistic *community model* subsequently. Simulation results show that PROPHET outperforms epidemic routing in terms of both delivery success rate and delay experience of messages. It also introduces far less traffic overhead than epidemic routing to manage forwarding.

The **MobySpace Routing** [LFC05] protocol exploits knowledge of mobility patterns of nodes to build up a high dimensional Euclidean space, named *MobySpace*, where each axis represents a possible contact between a couple of nodes and the distance along an axis (the coordinate is named *MobyPoint*) measures the probability of that contact to occur. Two nodes that have similar sets of contacts and that experience those contacts with similar frequencies are close in the MobySpace. Nodes that have very different sets of contacts instead, or that experience the same contacts but with very different frequencies, are going to be far in the MobySpace. The best forwarding node for a message is the node that is as close as possible to the destination in this space. This in fact improves the probability that the message will eventually reach the destination. Obviously, in this *virtual contact space* just described, the knowledge of all the axes of the space also requires the knowledge of all the nodes that are circulating in the space. This full knowledge, however, might not be required for successful routing. An alternative virtual space can be defined with a lower dimension, under the condition that the visits of nodes to particular locations can be tracked and that nodes have certain regularity in visiting a certain set of locations. Hence, each axis is chosen to represent a particular location and the distance along the axis is the likelihood of a node to visit that location. Nodes that have similar probabilities of visiting a similar set of locations are reasonably more likely to encounter each other than nodes that have very different probabilities in these respects. Obviously the efficacy of this virtual space as tool for making routing decisions can be limited if nodes change their habits too rapidly. In order for each node to construct its MobySpace of reference with

the information of all the other nodes, mobility patterns of nodes must be flooded over the network (through epidemic flooding) but this unfortunately leads to high protocol overhead. Nevertheless, optimization is possible by spreading only the main components of the mobility patterns (the components left are presumably negligible due to the power-law distribution of probabilities). According to the forwarding strategy of MobySpace routing, messages are sent towards the nodes that have mobility patterns successively closer to the mobility pattern of the destination. Hence a *similarity function* must be introduced to compare mobility patterns. Different alternatives have been investigated based on different distance metrics: *Euclidean distance*, *Canberra distance*, *Cosin angle separation*, and the *Matching distance* (for further details please refer to [LFC05]). MobySpace routing has been tested by considering mobility patterns resulting from real experimentations on WiFi environments. Locations in the MobySpace have been set to the positions of the Access Points that provided wireless access to the Internet in college campuses. Any couple of mobile nodes could communicate one another only in case they could see the same Access Point, i.e., in case they visited the same location (MobyPoint). Node movements were based on power-laws and each node had a mobility pattern defined by the distribution of $P$. $P(i)$ is the probability for the node to be in the location $i$.

$$P(i) = k\left(\frac{1}{d}\right)^{n_i} \tag{IV.15}$$

$n_i$ is the preference index of the location $i$; $d$ is the exponent of the power law, and $k$ is a constant. MobySpace routing has been compared with other routing approaches: epidemic routing, wait-for-destination, random routing (the next hop is chosen at random), and message delays as well as route lengths have been contrasted. Simulation results show that when full mobility patterns are exchanged among nodes MobySpace routing outperforms the other algorithms with both Euclidean and Cosin Angle metrics. Canberra and Matching metrics perform better in case of partial mobility pattern spreading.

### 4.2.2. Shortest path-based Routing

The **Extremely Opportunistic Routing protocol (Ex-OR)** [BM03] differs from the above utility-based protocols in that it does not seek to find the best next-hop node for a message before transmitting it. In fact, it defers any decision about the best forwarder for a message after the time this has already been sent. The best forwarder is selected among the nodes that have received the message for being the one that can reasonably offer, in that particular moment, the best delivery opportunity to that message with respect to all the other nodes that have received it. ExOR assumes that each node in the MANET has complete knowledge of the transmission *loss rates* between any couple of nodes in the network and that it can estimate from these, given a particular message to deliver, the *Expected Delivery Cost* metric from any node of the MANET to the destination node of the message itself. The Expected Delivery Cost measures the closeness of a node to the destination node. It corresponds to the minimum cost of delivery (with respect to different alternative paths) from that node to the destination node. The cost of delivery of a path is the sum of all the expected costs of delivery of the single links (inverse of the link's delivery probability) that compose that path. Further details on the Expected Delivery Cost metric metric can be found in [DCABM03] where a similar metric is presented named *Expected Transmission Count (ETX)*. To support the calculation of the Expected Delivery Costs from any node to any possible destination node in the network each node has to monitor the weakness of its local links over time and to accomplish *periodic flooding* of the measurements performed. After having the measures from all over the network, each node is able to build its own personalized list of the optimal forwarders for any destination node. The list is prioritized from the best forwarder to the worst. So, for example, node $A$ can have a forwarder list, for messages destined to node $D$, that establishes that the best next-hop node for forwarding is node $E$, and immediately after node $F$, then node $G$, and so on and so forth. When the highest priority forwarders are temporary not available, alternative forwarders with minor priority are chosen from the list. The ExOR protocol works as follows. Any source node wishing to deliver a number of packets to a destination node organizes groups of packets, i.e., batches, and broadcasts them one at a time (the protocol works on batches of

packets rather than on one single packets at a time to alleviate the overhead per single packet). Each packet is added a ranked list of forwarders (initially the same for all the packets of the batch), specifically, the list of those nodes having the lowest expected delivery costs towards the destination node of the batch. The list is worked out by the source node itself and is named *batch map*. All the nodes that it includes, from the best one to the worst, will help in turn forward the packet towards the destination. The list also includes both the source and destination nodes of the packet and specifies the highest priority forwarder node that has already forwarded that packet. After broadcast from the source node, the protocol selects the next forwarder node among those nodes belonging to the batch map that have received, even partially, the batch of packets. The selected forwarder re-broadcasts all the packets that it has previously received. Then, another forwarder is selected from the batch map to broadcast those packets it has received that have not been re-broadcast yet from the forwarder previously selected, and so forth until all the packets in the batch are re-broadcast. Specifically, each node that has received some packets of the batch and whose node $ID$ is included in the batch map of those packets, i) waits for the batch to be completely sent from the sending node; ii) waits for each higher priority forwarder node in the batch map to finish in turn re-broadcast the received packets; iii) starts broadcasting the packets of the batch that it has received and that have not been broadcast yet by any higher priority forwarder (each time the node overhears a packet of the batch being re-broadcast by a higher priority forwarder, the batch map for that packet is locally updated with the priority of the highest priority forwarder for that packet). The protocol well tunes the time instants when each forwarder node can start transmitting in order to avoid collisions. This kind of management lets different packets simultaneously follow different routes towards the destination node. By the means of long range radio signals, which are notoriously high lossy, some packets may reach nodes very close to the destination and thus experience a very few-hop path. On the other hand, unlucky conditions of the transmission means may cause some other packets only reach very close-apart nodes. However, this does not cause retransmission of the same packets (like in traditional routing protocols in case the target next-hop is not reached) and the subsequent broadcast for those packets will simply start from those same receiving nodes even

though not optimal. Each receiving node will retransmit the received packets following the order in the batch map. The destination node, when on its turn, will only broadcast the batch maps of the packets without any data. When all the forwarders will have re-broadcast their packets, a new round will be started again and another one after that, until $90\%$ at least of the batch's packets will have arrived to destination. All the missing packets will be then transmitted via a traditional routing protocol. A key issue for the protocol is clearly building up the list of forwarder nodes for a batch. The protocol has been tested in a real environment with $38$ fixed nodes, all $802.11b$-capable, spanning a $6$ $km^2$ wide area. Results show that ExOR prevents unnecessary retransmissions with respect to traditional ad hoc routing protocols and also increases the end-to-end throughput. The throughput gain varies with the number of nodes between source and destination and is nearly $35\%$ when the source and destination nodes are few-node apart from each other, and by a factor of $2$ to $4$ when farther. By transmitting each packet fewer times, ExOR also causes less interference for the other users of the network with respect to traditional routing protocols. Given the need for link state flooding, ExOR probably performs better when the network is almost-completely connected with only occasional disconnections and partitions.

The **Shortest Expected Path Routing protocol (SEPR)** [TZZ03] focuses on two key aspects of the forwarding strategy, specifically, to whom forward a message and which buffer replacement policy to adopt. The buffer replacement policy decides the best candidate for overwriting among all the cache entries of a node when a new message arrives, needing storage, and no more room is available. SEPR relies on estimation of the *Expected Path Length*, a metric similar to the *Expected Delivery Cost* used in ExOR, to make forwarding decisions. Each node in the network is expected to maintain a *link probability table* that contains, for each of the nodes known in the network, the probability that a link towards that node holds. The link probability between any two nodes, say node $i$ and node $j$, is computed as follows.

$$P_{i,\,j} = \frac{Time_{connection}}{Time\ Window} \tag{IV.16}$$

$Time_{connection}$ is the total number of time units that the connection between node $i$ and node $j$ has held and the $Time\ Window$ is the total sampling time. This prob-

ability is updated each time the two nodes meet one another. Moreover, the entire link probability tables are exchanged between nodes during pair-wise contacts. This allows more recent estimates to update older ones at both nodes. Moreover, through pair-wise exchanges, fresh link probability tables eventually propagate all over the network. Once updated their local link probability table, nodes can work out the expected path lengths to all the other nodes in the network. Assuming $T$ is the random variable that models the total time needed for a message $m$ to reach the destination node $D$ from the source node $S$, the *Expected Path Length* between node $S$ and node $D$ is the expectation value of $T$,i.e., $E(T)$. In fact, it can reasonably be considered that the time spent to go from node $S$ to node $D$ is a good metric for the distance between those two nodes. Given $\Pi(t)$ as the probability distribution function for $T$, the expected path length is computed as follows.

$$E(T) = E_{path} = \sum_{t=1}^{\infty} t \cdot \Pi(t) \tag{IV.17}$$

In case of multi-hop path where the time spent to traverse each link is represented by independent random variables, say $T_i$, the expected path length is computed as follows.

$$E_P(T) = \sum_{i=1}^{k-1} E(T_i) \approx \sum_{i=1}^{k-1} \frac{1}{p_{i,\ i+1}} \tag{IV.18}$$

The time needed to traverse a single link can reasonably be assumed to be inversely proportional to the probability for that link to hold. When multiple path are possible, the best is obviously the one which is fastest to traverse, i.e., the *shortest* path, as assumed here. It can be calculated with the Dijkstra algorithm. A message $m$ destined to node $D$ is forwarded from node $A$ to node $B$ during a pair-wise contact in case the expected path length from node $B$ to node $D$ is less than both the expected path length from node $A$ to node $D$ and the residual time-to-destination associated to the message itself (see below the *effective path length*). Cache updates are necessary whenever a new message arrives to a node and its cache is already full. To manage these updates, one more metric can be introduced named *effective path length*. It is associated to each message stored in the cache of a node and gives a rough idea of the residual time-to-destination for that message. So, when a new message arrives, the message to overwrite in the

full cache is the one with the lowest effective path length because this message is the one which is more likely to have already reached the destination. Hence, during pair-wise contacts nodes contrast the lists of their respective messages. If a node, say $A$, stores a message $m$ destined to node $D$ and realizes that the expected path length between the encountered node, say $B$, and node $D$ is shorter than the effective path length of the message, then it forwards the message to node $B$ and updates the effective path length, $EPL_m$, for message $m$ in cache as follows.

$$EPL_m = \min(EPL_m,\ E_{path}(B,\ D)) \tag{IV.19}$$

The SEPR protocol has been tested via simulation using a *City mobility model*, the same used in [DFL01] to allow easier contrasts. Simulation results show that SEPR allows $35\%$ increase in the success delivery rate with respect to epidemic routing as well as $50\%$ decrease in resource costs.

The **Practical Routing protocol** [JLW05] also relies on the prediction of the shortest path to the destination node of a message. The metric that it proposes minimizing, to define the shortest path, is the end-to-end delay to the destination node of the message. It is called *Minimum Estimated Expected Delay (MEED)* and resembles the *Minimum Expected Delay (MED)* introduced in [SFP04]. A completely connected graph is assumed to be built over the network. Each edge linking a couple of nodes represents a contact within the two nodes and is assigned a weight that depends on statistics on the connection and disconnection intervals between those nodes. The weight represents the minimum expected delay until that contact occurs. Both contact and inter-contact times experienced by nodes are appropriately composed and collected in a sliding history window at each node and then used to build up the routing table of the nodes. Infrequent contacts lead to high minimum expected delay because long time must be waited before they occur. Hence, they are not likely to be included in the shortest paths. The protocol re-computes the routing tables whenever a new contact occurs (per-contact routing) thus assuring that each routing decision is made with the most recent information. Moreover, a cost of zero is assigned to any contact that is newly available. Zero-cost links are sorts of "short circuits" in the network graph and are very likely to be included in the shortest paths. The routing tables are also recomputed before any mes-

sage is forwarded to insure that the most recent information is exploited to make forwarding decisions. However, this causes many resources to be spent and additional delay to be introduced before a link can be used. This approach does not scale very much in large networks. When the weights of the links change fastly and frequently loops are possible so that, when the packet reaches one node, the routing directs it back the way it has come. Short-circuit routing further aggravates this problem but mitigation is possible by adding some hysteresis. Variations in the local link state table of each node are propagated to all the other nodes in the network in an epidemic fashion through pair-wise contacts. Weight variations are propagated only in case they are greater than $5\%$ or when a new contact has occurred (so some negligible updates are actually suppressed). Unfortunately this solution consumes plenty of memory at nodes to store the complete network topology and causes great effort to be spent in merging topology information coming from different nodes. The protocol has been tested via simulation over the same DTN simulator used in [SFP04]. Nodes have been moving according to the mobility model resulting from the real traces presented in [HKA04]. Results show that Practical Routing has much better performance than Epidemic Routing when nodes have limited storage capacity. The delivery ratio and the latency improve when increasing the buffer size. The protocol overhead has a quadratic increase with the number of nodes of the network $O(n^2)$.

### 4.2.3. Context-based Routing

The **Interrogation-Based Relay Routing (IBRR)** [SBRJ02] protocol has been conceived for ad hoc satellites networks and specifically for *ad hoc Scientific Earth Observing (SEO) satellite networks* which are composed by both satellites and earth stations. Satellites in these networks have no pre-defined topology and may have different altitudes, orbits, and link capacities. Since inter-satellite links can only offer intermittent connectivity, each time a satellite node needs to forward a message to another satellite node it searches the best next hop among all the available neighbours. This is done by evaluating the context information collected during pair-wise contacts between satellites. Context information include: i) spatial location and orbital information, ii) bandwidth of the inter-satellite link between the

neighbours, iii) relative velocity/mobility, iv) vicinity of the neighbour to other satellites and ground stations, v) capability of the near satellite, vi) data transmission time. During a pair-wise contact between satellites they request each other their context information as well as the context information of their neighbours thus performing a so called *one-hop look-ahead*. The one-hop look-ahead gives each node awareness of the surrounding network topology. When a node needs to send a message, it selects the best next hop available by exploiting the information gathered. In case no next hop is available at the moment, IBRR performs *optimistic forwarding* [CKV01] and buffers the message locally waiting for a new forwarding opportunity to be available.

The **Context-Aware Routing protocol (CAR)** [MHM05] uses a classical routing protocol to provide message forwarding inside the connected clouds of a partitioned ad hoc network. Specifically, it makes use of a proactive approach, so, each node of the network holds its own routing table that binds a next-hop node to each known (potential) destination node located in its partition. The proactive routing protocol is able to find connected paths between a source node and a destination node both located inside a connected ad hoc cloud. The nodes which are located in other partitions of the same network instead, are not reachable through a classical routing protocol. Nevertheless, there may exists a node in the same partition of the source/forwarder node that is likely to get within reach of the destination or to meet another node which is likely to reach the destination node. CAR focuses on expanding the classical routing table of nodes to support forwarding across disconnected ad hoc clouds. It adds a couple of columns to the routing table. They give the best carrier towards a certain destination host and the corresponding delivery probability that measures the likelihood of successful delivery to destination through that carrier. Each node in the network is in charge of producing its own delivery probabilities towards each known destination host. Delivery probabilities are exchanged periodically together with classical routing information so that, eventually, each node can compute the best carrier for each destination node (i.e., the node with the best delivery probability towards that destination). When sending a message towards a certain destination, a node forwards the message either to the corresponding next hop, in case the destination is within the same connected ad hoc cloud of the source/forwarder node, or to

the best next-hop in case no connected path to the destination exists. Among the context attributes for the election of the best carrier are, for example, the residual battery level, the rate of change of connectivity, the probability of being in the same cloud of the destination, the degree of mobility. When the best carrier receives a message for forwarding, it stores it in a local buffer and will eventually forward it to the destination node when met, or alternatively to another node with a higher delivery probability. Since different context attributes can be defined, it needs defining a way to combine them into a single delivery probability. CAR makes use of the *multi-attribute utility theory* to produce the *utility function* of a set of attributes $(X_1,\ X_2,\ \ldots,\ X_n)$ with values $(x_1,\ x_2,\ \ldots,\ x_n)$. The utility function can be derived either with a simple additive function or a weighted additive function depending on the attributes being *mutually preferentially independent* or not. The formulas for calculating the utility functions in both cases follow.

$$\begin{aligned} U(x_1,\ x_2,\ \ldots,\ x_n) &= \textstyle\sum_{i=1}^{n} U_i(x_i) \\ f(U(x_1,\ x_2,\ \ldots,\ x_n)) &= \textstyle\sum_{i=1}^{n} a_i(x_i) \cdot w_i \cdot U_i(x_i) \end{aligned} \tag{IV.20}$$

$w_i,\ i = 1,\ \ldots,\ n$, are the *significance weights* and reflect the relative importance of each attribute whereas $a_i(x_i),\ i = 1,\ \ldots,\ n$, are variable coefficients that increase or decrease significance weights over time depending on the particular value assumed by an attribute, on its predictability (stronger or weaker correlation), and on its availability (how recent is the arrival of fresh updates). As is stated above, delivery probabilities are exchanged periodically among nodes in the network to guarantee that forwarding decisions are always supported by fresh information. Moreover, between successive exchanges delivery probabilities are also updated by means of predictions to limit bandwidth consumption. Prediction techniques make use of Kalman filters. CAR has been tested via simulation on OMNET++ and contrasted with flooding and epidemic routing. A Group Mobility model has been assumed for nodes in the network and the attributes considered have been i) rate of connectivity change and ii) probability of the node being located in the same cloud as the destination. Simulation results show that epidemic routing outperforms CAR in terms of both delivery ratio and delay while CAR outperforms simple flooding. However, CAR shows far higher scalability than epidemic routing

in that the protocol overhead is approximately constant regardless of the node buffer size.

### 4.2.4. Signal strength-based Routing

**Signal strength assessment Based Routing enhancementS to OLSR (SBRS-OLSR)** [SBS+03] have been introduced to improve the performance of the OLSR routing protocol especially in case of wireless networks characterized by highly mobile nodes and instable routes. SBRS-OLSR monitors at each node the signal strength of the outgoing links towards one-hop neighbours. For forwarding decisions links are selected based on their quality measured in terms of signal strength. Hence, when a mobile node leaves its previous position and becomes gradually farther from its previous neighbours, the next hops resulting from its routing table are substituted by the incoming next-hop neighbours allowing stronger connections. SBRS-OLSR introduces a new metric called *affinity* that quantifies the likelihood of a link between a pair of nodes to hold. First, each node, say $n$, maintains the histories of the averaged SNR values towards each of its neighbours in circular linked arrays, say $snr_{n,m}$, $m$ indicating the specific neighbour node. Then, for each neighbour node $m$ the *average rate of change* $r_{n,m,ave}$ of the $SNR$ is evaluated as follows.

$$r_{n,m,ave} = \frac{1}{N} \cdot \sum_{i=1}^{N-1} \frac{snr_{n,m}[current] - snr_{n,m}[(current+i) \bmod N]}{t[current] - t[(current+1) \bmod N]} \qquad \text{(IV.21)}$$

$N$ is the size of the array $snr_{n,m}[k]$, $k = 0, \ldots, N-1$, which stores the $SNR$ values to the neighbour $m$ sampled at different time instants $k$. $t[k]$, $k = 0, \ldots, N-1$, is the timestamp array that contains the sampling instants of the $SNR$ values stored in the $snr_{n,m}[k]$ array. The affinity between the two nodes $n$ and $m$ is then calculated as follows.

$$a_{n,m} = \begin{cases} high & if\ r_{n,m,ave} > 0 \\ \frac{snr^{thresh} - snr_{n,m}[current]}{r_{n,m,ave}} & otherwise \end{cases} \qquad \text{(IV.22)}$$

$snr^{thresh}$ is the lower-bound limit for the $SNR$ of a link. If the $SNR$ of a link be-

tween two nodes is below this limit, the link is assumed to be disconnected. The neighbour table built by the OLSR routing at each node $n$ is added a $SNR$-history array field for $snr_{n,m}$, as well as the corresponding timestamp array, and a $SNR$ smoothing array for averaging the instantaneous $SNR$ values. Node $m$ is a good candidate to be selected as $MPR$ for node $n$ if the link between node $n$ and its neighbour node $m$ is expected to last long. This is possible when the nodes are approaching each other and thus, their affinity is high and the $SNR$ is above the threshold. The route selection algorithm selects the route towards the destination that passes through the $MPR$ (as the next hop from the given node) that provides the higher minimum affinity along the route to the destination. Performance results obtained by experiments in a real environment confirm the effectiveness of the protocol.

### 4.2.5. Location-based Routing

The **Spraying Protocol** described in [TR01] manages delivery of messages towards highly mobile nodes by assuming that nodes that go away from their current position can be reached, for some time at least, in the nearby of this position. Hence, the best way to reach the actual destination of a message that is likely to have moved from its previous position is to deliver the message to the nodes located in the neighbourhood of the destination's last known position. According to the network model assumed in the Spray Protocol, two kinds of nodes exist: *switches* and *endpoints*. Switches have routing functions whereas endpoints may be source or destination of messages. At each location an endpoint associates to a switch that means that that switch is its default router for message delivery. A Location Manager (LM) is assumed to be present in the network. It collects position information about the nodes of the network in a database. Specifically, the LM stores for each endpoint the corresponding selected switch (endpoints and switches are said to be *affiliated*). So, each time a node moves to a new position, it sends a *Location Update* to the Location Manager giving it knowledge of its new affiliation. Whereas, each time a source node wishes to send a message to a destination node, it first sends a *Location Subscribe* to the Location Manager to ask for the last affiliation of the destination node. The Location Manager an-

swers with a *Location Information* giving the destination last known affiliation and also a couple of parameters: *depth* and *width* respectively. Then, the source node unicasts the message towards the destination' last known affiliation (the route is found through a classical reactive approach), and the receiving node broadcasts the message to its neighbours and to its neighbours' neighbours. *Width* gives the number of levels of neighbours to which the traffic should be multicast. The last group of neighbours that receives the message is *depth* hops apart from the node that first performed broadcast (the switch). The computation of the parameters width and depth is performed by the Location Manager by taking into consideration the history of the affiliation changes for each endpoint. If an endpoint frequently changes its affiliation, it is highly mobile, thus the broadcast' width and depth should increase. The efficacy of the protocol has been tested via simulation. The simulation tool had been written from scratch using the C++ language. The only mobility model considered for nodes was the Extended Random Walk. The protocol outperforms reactive protocols when the mobility of nodes increases.

The **Mobile Relay Protocol (MRP)** [NPB03] has been conceived to integrate pre-existing ad hoc routing protocols and manage message forwarding when no route towards the destination node of a message is found and the application that has generated the message can tolerate some form of extra delay. Messages that can be forwarded in opportunistic fashion are assigned two parameters: $h$ and $d$. $h$ represents the upper bound limit for the number of times the message can be relayed, i.e., the maximum number of relays that it can visit. Each time the message reaches a new relay node, $h$ is decreased by one so as to correspond to the residual number of relays that it is allowed to visit from then on. On the other hand, $d$ represents the upper bound limit for the length of a multi-hop path towards the destination node according to a traditional routing protocol. Therefore, a message can overall traverse $h$ hops over a non-connected path and $d$ hops over a connected path. The MRP protocol works as follows. Assume that an MRP layer is added on top of the network layer (where a classical routing protocol is implemented) so as it can be involved in the forwarding of messages that are generated by delay-tolerant applications whenever the traditional routing protocol fails delivery. Assume also that messages are marked when they are treated at the MRP-layer so as they can be treated differently from the others. Whenever a node

manages delivery of a message that has not been MRP-treated previously (either an own one or one that has just been received from another node), it looks for a route by means of a table lookup (proactive approach) or a route discovery (reactive approach). If a route is found towards the destination that is less than $d$ hops long, the node delivers the message throughout that route. Otherwise it attempts a local broadcast of the message to its immediate neighbours hoping that some of its neighbour can find a way to destination. It also stores the message locally and enters a so called *relaying state*. In the relaying state the node periodically checks if a route towards the destination of the message exists that is less than $d$ hops long. If so, the message is delivered immediately. When a node receives a message from another node and this message has already been MRP-treated (i.e., the message has already been broadcast once), it looks for a standard multi-hop connected path towards destination and it delivers the message in case the route is found. Otherwise the node enters the *storing state* and simply stores the message. If the message already exists in the node buffer, the new message is dropped. If the buffer is full, then the oldest message in the buffer is replaced with the new message. The message that is discarded is sent to an immediate neighbour after having decreased its associated $h$ parameter, only in case the result is not $0$. If $h$ equals $0$ instead, the message is dropped. The immediate neighbour for relaying is chosen randomly. Once having stored the message, the node enters the *relaying state* and periodically checks availability of routes towards destination of all the messages being relayed. This protocol relys on the presumption that even though a node has not a route towards a particular destination, it is likely that one of its immediate neighbours can have it or is going to have it soon. Thus, messages are not diffused everywhere but locally and one only broadcast is attempted whereas further relays are involved only in case of buffer overflow. By limiting message diffusion, the MRP protocol limits the bandwidth utilization. It also manages a bounded storage capacity. The efficacy of the protocol has been tested via ns-2 simulations under a variety of mobility models (random waypoint, soccer player, and homing pigeon movement models). The protocol has been tested in combination to the DSDV routing protocol.

## 4.3. Infrastructure-based Routing

In Infrastructure-based routing, the network includes some infrastructure to perform delivery of messages. Infrastructure consists of special fixed nodes, base stations, which are sparsely deployed all over the network and act as message collectors. Base stations offer high capacity and robust exchanges to the mobile nodes in the nearby. Moreover, they have high storage capacity to collect data from many nodes passing by them. In infrastructure-based routing a source node wishing to deliver a message keeps it until it comes within reach of a base station, then forwards the message to the base station. The base station stores the message waiting for the the destination node to pass by. When the destination node is within reach of the base station, this delivers the message. Two variations of the protocol are possible. The first one works exactly as described above. Only node-to-base-station communications are allowed, as a result, messages experience high delays. However, this solution is very energy efficient because mobile nodes are completely relieved from the forwarding workload. A second version of the protocol allows both node-to-base-station and node-to-node communications. This means that a node wishing to send a message to a destination node delivers the message to the base station directly if within communication range, otherwise it delivers the message to a near relay node that eventually will forward the message to the base station in case it will pass by it. This second protocol is less energy efficient than the previous one but guarantees lower delays. Despite allowing energy saving at mobile nodes, infrastructure-based routing is nevertheless a highly expensive solution due to the infrastructure costs.

**Infostations** [GBMY97], [IR00] are base stations which offer wireless coverage to limited, isolated geographic areas. In addition, infostations have the potential to support high bit-rate wireless connections to the mobile users who get within their communication range. According to the infostation communication paradigm, data exchanges may occur only in proximity of the infostation itself (node-to-infostation, infostation-to-node). Services like web access can thus be enjoyed wherever an infostation is present, e.g., at shopping malls or along sidewalks (walk-through scenarios), along highways and roadways (drive-through scenario), in an airport lounge, in a classroom or conference room (sit-through

scenario). Whereas the goal of cellular industry has been coverage anytime and anywhere, the Infostation model offers some sort of many-more and many-where coverage. In fact, Infostations are only able to offer services locally to those mobile users who pass by them. The user satisfaction is made possible by the high capacity of communications that Infostations can support and that makes possible large data exchanges in very little time, presumely corresponding to the total time the mobile user is within range of the Infostation itself. Services generally provided by the Infostations concern Web Access and thus the downstream channel is assigned much more capacity then the upstream channel. The main issue with Infostation services is intermittency of connections which may lead to the decrease of user satisfaction. So, different Infostations may organize in clusters that are controlled by a cluster controller. Instead, no horizontal coordination between Infostations is managed. Whenever a download is ongoing towards a mobile user through a particular Infostation, depending on the mobile user speed and direction, the cluster controller may decide to trigger some prefetch of data to another Infostation which will probably be the next one to be visited by the mobile user. This can help guaranteeing continuity to some degree.

The **Shared Wireless Infostation model (SWIM)** [SH03] (see Section 3.2.7) is an example of infrastructure-based routing where both node-to-node and node-to-Infostation communications are possible.

## 4.4. Compulsory Routing

Compulsory protocols are protocols that affect the natural motion of some or all the hosts of the network to ensure the correct protocol execution. In the protocol described below nodes in the network adjust their position and motion when a new message is generated and needs forwarding. It provides appropriate *connectivity* "on demand" towards the destination node to support message delivery.

The work in [LR00] proposes a proactive approach to support communications in disconnected ad hoc networks. Specifically, nodes in the network collaborate to build up a connected path between any couple of nodes willing to communicate. In this process, the sender node and the nodes elected as relays actively modify

their trajectories to create a connected path between sender and receiver. Two algorithms are proposed. The first algorithm relies on two assumptions: i) that the moving trajectories of all the nodes in the system are known, ii) that the hosts may reach so high speeds that the time needed for a host to deviate from the original trajectory towards a relay host is negligible. Alternative trajectory arrangements are taken into account before delivery of a message and eventually the one that minimizes the overall movements necessary for relaying is chosen. The *Optimal Relay Path* between two hosts $S$ and $D$ corresponds to the shortest time strategy to send a message from one host to another (taking the least time) and is given by a list of tuples of <*host, path-to-the-next-host*>. When the list is complete, the sender node just approaches the first node of the list, sends the message to it, then goes back to its original trajectory. The subsequent relays behave similarly. This approach is tricky in that trajectory modifications to allow delivery of a message have the potential to interfere with the concomitant transmission of another message. So, a host may not be found if it has just gone on a relay task, a host might have to relay multiple messages in different directions at the same time, and finally, it may happen that all the hosts decide to approach each other to deliver messages at the same time and in a circular fashion leading to deadlock. The second algorithm relieves the assumption of complete a priori knowledge of the hosts' trajectories and introduces location updates to inform of the movement of hosts. Each host is assumed to move around within a region, named *scope* of that host, and to know which hosts keep track of its location. When a host leaves its current scope it sends location updates to all the hosts which are aware of its position. The optimal path between any two nodes is on the minimum spanning tree of the network. It is obtained by modeling the network as weighted graph with hosts as vertices and physical distances between the nodes as edge weights. The minimum spanning tree of the graph contains the shortest edges in the graph that provide full connectivity in the graph. The neighbours in the minimum spanning tree provide the communication routes for messages. Each host has the responsibility of updating its location by informing all the hosts in the minimal spanning tree that are included in its scope. Thus, when a host leaves its scope, it only needs to inform its neighbours in the minimum spanning tree. It is clear that there is a trade-off between the size of the host's scope and the frequency of its location

update messages. This method for relaying messages is quite heavy and adds to the node considerable working load. To limit the impact on the working time, hosts delay the delivery of messages and accumulate messages up to a point thus avoiding immediate preemption on running tasks. It has been demonstrated that these protocols work fine when the network is almost completely connected and the distance between two hosts is slightly larger than the transmission range so as the hosts need to move small distances to relay messages. Valuable application scenarios are military missions and emergency relief where teams move relatively close to one another and communication is done purely on the ad-hoc network.

## 4.5. Carrier-based Routing

In carrier-based routing special nodes take on the role of mobile data collectors. They move around in the network following pre-determined or arbitrary routes and gather messages from the nodes they pass by. These special nodes may be referred to as *carriers*, *supports*, *forwarders*, *MULEs*, or even *ferries*. They can be the only entities responsible for delivery of messages, when only node-to-carrier communications are allowed, or they can simply help increase connectivity in sparse networks and guarantee reachability to isolated nodes. In this latter case, delivery of messages is accomplished both by carriers and ordinary nodes and both node-to-node and node-to-carrier communication types are allowed. Introduction of special mobile relay nodes in the network is advantageous from the energy efficiency standpoint. When only node-to-carrier communications are possible, ordinary nodes are relieved from the routing workload. Moreover, node-to-carrier communications are generally short range radio communications and thus cheaper. Nevertheless, this communication system has a drawback that is the high delay experience of messages. Systems that adopt carrier-based routing must be highly delay-tolerant like, for example, wildlife tracking systems. To decrease the average delay of message delivery, also ordinary nodes may take on the role of forwarders, but the energy saving worsens. Mobile carriers add scalability to the network so, message delivery is granted also when increasing the total number of nodes, and without increasing the routing overhead indeed. However, when increasing the number of nodes, the delay experience of messages generally in-

creases too. To limit the delay increase new carriers should be added. Utilization of multiple carriers is also recommended for the sake of robustness: a single carrier responsible for delivery of messages is necessarily single-point-of-failure for the network. Nevertheless, when multiple carriers are present in the network, some tricky issues arise from the definition of the optimal paths to follow to their coordination. Eventually, carriers transmit collected data to a final base station. This may happen in different ways:

1. By direct transmissions making use of long range radio: this allows optimal delay, but also needs special instrumentation and hence is highly expensive.
2. By waiting to be near to the base station and transmit over a short range: this solution is cheaper than the previous one but also increases the delay.
3. By diffusing the data collected among the other carriers in an epidemic fashion. Eventually data will reach the base station. The description of some carrier-based routing protocols follows.

The **Snake Protocol** [CNS01] and the **Runners' Protocols** [CNP+01] are said to be semi-compulsory protocols because they select a very little subset of the mobile nodes of the network to act as *supports* for the other nodes. Supports move around in the network area and, as a result, create delivery opportunities. According to the Snake Protocol supports coordinate among them. They move in a snake-like sequence following the random walk of the first node, i.e., the snake's head. Hence, the motion of the sequence of supports is unpredictable. Nevertheless, it spreads all over the network area. All the supports always hold pair-wise contacts among them. Whenever one of the supports gets within communication range of a source node, the source node is informed that it may send its messages to the support. Once received the messages from the source node, these are stored somewhere within the support structure (e.g., at all the supports). Similarly, when a receiver node comes within communication range of a node of the support structure, the receiver is notified that there is a message for it and the message is then forwarded to the receiver. Both analytical and experimental analyses have shown that a very small number of supports is really needed for efficient communication. However, the protocol is not robust because it is resilient to the

fault of just one node (i.e., the fault of one of the supports). The Runners' Protocol is an enhancement of the Snake Protocol where each node of the supports moves independently from the others and according to a random walk mobility model. Supports' motion spreads all over the network area. Whenever two supports come within communication range of each other they exchange the data gathered from the source nodes. The runners' protocol outperforms the snake protocol because it both achieves better average message delay and higher delivery rate, and it also requires smaller storage capacity at the nodes. In addition, it is more robust because is resilient to the fault of up to the total number of supports minus one.

Another semi-compulsory protocol similar to the Snake and to the Runners' protocols introduces in the network place the so called **Virtual Mobile Nodes** [DGL+04] to allow execution of distributed algorithms. They move around in the network over pre-determined or at least predictable routes and provide services to the client nodes that are gradually met. Real nodes (as opposed to virtual nodes) move independently from virtual nodes and in an arbitrary and unpredictable manner. When a source node has a message to send it waits for a virtual mobile node to pass by and then delivers the message to the VMN. Each node is assumed to know its own position. Moreover, a global timing system is assumed to be present so as each node can calculate the time and position of the next meeting with the VMN. After sending the message to the VMN, the message is stored inside the VMN and finally delivered when the VMN approaches the destination node. A robust and reliable implementation for the VMNs has been proposed in [DGL+04] that makes use of *mobile agents* moving from real node to real node. At different times the role of the VMN is thus taken on by different real nodes. Moreover, the state of the VMN is replicated on a set of nodes so as to provide a more robust service. The VMN abstraction consists of some real nodes included in a circular area named Mobile Point. The centre of the Mobile Point coincides with the pre-planned location of the VMN at time $t$. While moving over the pre-defined path, new real nodes enter the Mobile Point whereas others leave it. When a message is sent to a node in the Mobile Point, it is replicated and sent through a local broadcast also to all the other nodes in the Mobile Point structure, thus, every real node that resides in within the Mobile Point replicates the state of the virtual node. Also the delivery from the source node to the first node of the Mobile Point

as well as that from the VMN to the destination node of the message occurs via local broadcasts. Obviously, VMNs can only exist if real nodes are present in the network place. This solution has therefore better performance in dense MANETs. Communications between virtual nodes and real nodes occur over short distances and result in good energy efficiency, nevertheless, high storage capacity is needed at all the nodes to maintain the state of the VMN. Computation of the optimal path for the VMN with the purpose of reducing the message delay time has not been addressed in [DGL+04].

**Smart-tag based data dissemination** in sparse wireless sensor networks is a system for data collection and dissemination based on utilization of mobile carriers [BLB02]. The overall system consists of few sensor nodes scattered over a wide area to perform environmental monitoring and some fixed nodes placed in strategic positions, instrumented for displaying the information gathered by the sensor nodes. Mobile carriers are introduced indeed to physically transport information from the sensor nodes to the remote fixed nodes. They consist of *Smart Tags* that are small radio-capable devices with some limited storage capacity. The application scenario considered for this system is a wide national park with fixed weather stations, sparsely located all over the park area, performing data monitoring and fixed electronic display boards (hereafter referred to as access points) located along the routes generally followed by hikers. Display boards are able to display the weather information collected at the weather stations. The system also includes small mobile devices, i.e., smart tags, which act as data carriers. Specifically, hikers are equipped with smart tags and can retrieve data from weather stations when in the nearby as well as update the data displayed at boards along the paths when they pass by them. Electronic boards are also capable to send information to the smart tags when within reach so as smart tags can update farther boards. Data exchanges occur from weather stations to smart tags and both from smart tags to access points and from access points to smart tags. Communications are allowed neither among static nodes nor among smart-tags. Each single data exchange occurs like in the epidemic protocol. The data generated at source nodes is assigned a source identifier and a timestamp. When a smart tag passes by a data source, a connection is established and the data is passed to the smart tag. Then, when the smart tag passes by an access point, the data is first sent from the

smart tag to the access point then, updates are sent from the access point to the smart tag. A buffer replacement policy is accomplished in case there is no room in the smart tag memory for the new data. Possible policies are:

1. *Sticky*: only the data that is least recent than the data stored at the fixed node is replaced, starting from the oldest data;
2. *Round Robin*: the buffer of the smart-tag is completely replaced with the data from the fixed node. Data is picked from the fixed node in a round robin order;
3. *Random*: the buffer of the smart-tag is completely replaced with the data from the fixed node. Data is picked from the fixed node at random.

Simulation results show that data diffusion is more efficient when increasing the number of smart-tags and the length of paths followed by the hikers. As regards the buffer replacement policy, the round robin and random policies outperform the sticky policy instead the difference between Round Robin and Random is minimal. Smart-tag based data dissemination allows considerable energy saving because it only makes use of short range radio transmissions. To guarantee complete data exchange between the devices during a contact interval, connection should be available for at least some minimum required time. Anyway no further investigation on this issue has been addressed yet.

The **Data-MULE system** [SRJB03] [JSB+05] relies on the same basic concepts of the smart-tag system above but it concentrates on data retrieval from sparse wireless sensor networks rather than on data dissemination. It consists of a three-tier architecture:

1. The lower level is occupied by the sensor nodes that periodically perform data sampling from and about the surrounding environment.
2. The middle level consists of mobile agents named *Mobile Ubiquitous LAN Extentions*, or MULEs for short. MULEs move around in the area covered by sensors to gather their data, which have previously been collected and temporarily stored in local buffers. Data MULEs can be for example people,

animals, or vehicles too. They move independently from each other and from the sensor positions by following unpredictable routes. Whenever they get within reach of a sensor they gather information from it.

3. The upper level consists of a set of wired Access Points (APs) and data repositories which receive information from the MULEs. They are connected to a central data warehouse where the data received is synchronised and stored, multiple copies are identified, and acknowledgments are managed.

Sensor nodes are supposed to be immobile and continuously awake waiting for a MULE to pass by for sending data to it. Sensor-to-MULE transmissions make use of short-range radio signals and hence do not consume too much energy. While moving around, when the MULE eventually passes by any AP deployed in the area, it transmits the collected sensors' data to it. Thank to the short-range radio exchanges, the Data MULEs' architecture is an energy-efficient solution for data gathering in sparse sensor networks (if compared to solutions based on the introduction of base stations to cover the entire area to monitor and also to solutions based on the introduction of a high number of sensor nodes to form a dense, entirely connected, sensor network). It also guarantees scalability and flexibility against the network size. Unfortunately, this solution has a couple of limits, both depending on the randomness of the MULEs' motion. Firstly, the latency for data arrival to the APs is considerable because some time elapses from the sampling instant to the moment the MULE takes the data, and then till the time the MULE actually reaches the AP and delivers the data to it. The second drawback is the fact that sensors have to continuously wait for any MULE to pass and cannot sleep. This leads to energy wastage. The scalability of the architecture has been investigated both analytically and via simulation. A two-dimensional grid has been taken as reference scenario with the APs uniformly distributed on the area and the sensors randomly distributed. MULEs have been assumed to move independently one another, each following a Discrete Random Walk mobility model. No data exchange has been assumed to occur among the MULEs and, finally, time synchronization has been assumed to be present among sensors and MULEs. Results show that the system parameters need adjustments when increasing the size of the grid, i.e., the size of the area to be monitored. Firstly, since the frequency

of the visits to the sensors by MULEs naturally decreases, an increase in the buffer size of the sensors is needed to prevent data loss. The latency experienced by the data monitored increases too. The increase is linear with the grid size. An increase in the number of MULEs can alleviate these effects nevertheless. Another effect of increasing the grid size is the decrease in the frequency of the visits to the APs by the MULEs. This leads to a further increase in the latency of data and to the need to increase the buffer size at the MULEs to prevent data loss. An increase in the number of APs can help alleviate the above effects. In conclusion, the number of MULEs can be traded for the size of the sensors' buffers whereas the number of APs can be traded for the size of the MULEs' buffers.

The Data MULE communication system has recently been exploited in **Underwater Sensor Networks** [VKR+05] deployed to monitor and model the behaviour of the underwater ecosystems. These sensor networks gather physical variables such as water temperature, pressure, conductivity, turbidity, and also pollutants' concentration. Moreover, underwater sensors may collect images to measure visible changes in the deep underwater environment or even to classify species. The network consists of both static nodes and mobile nodes. Static nodes are sensor nodes that perform data collection and storage. They are extremely power efficient because have little energy available and they are not easily rechargeable. Mobile nodes are *Autonomous Underwater Vehicles (AUVs)* which are responsible for data collection from the sensor nodes. They navigate the network to get within communication range of sensors to collect data from them. AUVs require much more power than sensor nodes because they navigate the sensor network however, they are quite easily rechargeable. Static nodes are mostly in deep sleep mode and wake up every few seconds to determine if they are being signalled by mobile nodes in the nearby. Relieving static nodes from most of the communication and storage loads contributes maximizing the network lifetime. It has been found that the most efficient way for collecting data from an underwater sensor network is using a system capable of both optical and acoustic communications. Optical communications guarantee high data rate and high bandwidth but need line-of-sight to be established between the communication peers and can only cover short ranges. Acoustic communications on the other hand have the potential for higher transmission range but suffer from attenuation and reflections and

allow lower bandwidth thus needing a trade-off to be met between communication range and data rate. Due to the broadcast nature of acoustic communication, when an acoustic transmission is holding, any other node is prevented to transmit, even to signal an event. Nevertheless, an optical communication and an acoustic one may hold simultaneously. Hence it has been established that the optical communication system is used for short-range line-of-sight data transfers between sensor nodes and AUVs (data mules). These transmissions are aimed at downloading the stored data from the sensor nodes and uploading commands to them. As they may involve much data, a faster communication system is more appropriate to use. The acoustic system is instead used to signal events over long distances and transmit small amounts of data. Signalling an event allows the AUV to move to the area of interest, and may also trigger redeployment of the sensor network to concentrate on some important features in the environment. Acoustic communications are indeed particularly suitable for sensor node localization. In fact, the speed of sound in water is low enough to permit accurate timing of signals to determine the distance between nodes. Pair-wise node distances are then used to perform 3D localization. The tasks of the mobile node are to establish a tour of the network, locate each node in the tour, one at a time, and hover above each node to download the data optically. During this period of communication the mobile node may also upload data to the static node, for example to adjust its clock or to change the data sampling rate. The key challenges for underwater data muling are a) locating the first node of the sequence to visit, b) locating the next nodes of the sequence, c) controlling the hover mode (for the mobile node), d) accomplishing data transfer, and e) synchronizing clocks so that the data collected by the sensor network is consistently time stamped. Localization of the first node of the data muling tour starts by positioning the robot in the general area of the network. Given that the general location is known in GPS coordinates, the AUV can perform surface navigation guided by GPS to move toward the node. Once close the AUV descends to the optical communications range. At this point the AUV performs a spiral search to locate the node by making use of distributed localization algorithms built on top of acoustic ranging.

The **Message Ferrying Approach** proposed in [ZAZ04] targets both increase of data delivery and optimization of energy consumption in sparse MANETs. Extra

mobile nodes are introduced in the network, with the only function to offer a message relaying service. These nodes are named *Message Ferries* and move around in the network place where they collect messages from source nodes. Message collection may happen in two ways:

1. *Node Initiated Message Ferrying*: the ferry node moves around following a pre-defined and known path. Each node in the network has knowledge of the paths followed by active ferries. The node wishing to deliver a message moves towards the nearest ferry and when sufficiently close, forwards its messages. Hence, the source node changes its trajectory to meet up with the ferry. This may obviously cause some degradation in the performance achieved by those applications that are currently running on the node during route deviation. Therefore, the source node controls its trajectory towards the ferry while striving to balance between performance degradation in the running tasks and performance gain in data delivery (i.e., minimizing message drops).

2. *Ferry-Initiated Message Ferrying*: the ferry node, again, moves around following a pre-defined, default path. To let other regular nodes know its position with good approximation, the ferry periodically sends its position information via long-range broadcast signals. Any source node wishing to deliver messages sends a `ServiceRequest` to the ferry also via a long-range radio signal. The source node also includes its current position in the `ServiceRequest`. After having received the request from the source node, the ferry changes its trajectory to meet up with the source node. The source node periodically communicates `LocationUpdates` to let the ferry adjust its trajectory in order to meet it. The new trajectory of the ferry is computed with the aim to minimize message drops. When the ferry and the source node are close enough message exchanges occur by means of short-range radio signals.

In both cases each node is expected to have location awareness, i.e., to know its position as well as the position of the ferry/ies, for example through GPS receivers. The protocol has been tested considering only one ferry to be present and

active in the network and no details are given on the multi-ferry case and on the necessary coordination between ferries. Since data delivery is only provided by ferries (node-to-node exchanges are avoided to save energy), the single-ferry case has a single point of failure in the ferry itself. Tests have been conducted via ns-2 simulations with nodes moving according to the *Random-Waypoint Mobility* model and some variations of it, i.e., the *Limited Random-Waypoint (LRW)* model and the *Area-Based (AB)* model. Simulation results have shown that the protocol achieves better delivery rate than Epidemic Routing. It also outperforms Epidemic Routing from the energy efficiency standpoint though it generally leads to higher message delays. The *Node-Initiated Message Ferrying* approach shows better performance than *Ferry-Initiated Message Ferrying*. The protocol shows no dependence on node mobility since in fact it makes no use of message relaying at intermediate regular nodes.

**DakNet** [PFH02] [PFH04] is a very low cost asynchronous ICT infrastructure that provides connectivity to rural villages in India. It is a low power consuming, scalable, robust, and easy to use system that meets the requirements of the rural market. Rural villages are generally isolated and lack many services, for example, they have not local government offices. Hence, whenever villagers have to request a driver's licence, apply for a loan, file compliances, etc., they need to reach the nearest town with government offices. Moreover, they have no infrastructure for Internet access so they cannot access these services remotely. According to the DakNet project, kiosks are built up in villages and equipped for digital storage and short-range wireless communications. Periodically, some Mobile Access Points (MAPs) pass by these villages' kiosks and exchange data with them wirelessly. MAPs are portable storage devices and radio-capable, mounted on and powered by physical means of transportations such as buses, motorcycles, or even bicycles. MAPs can upload any sort of request or data stored at the kiosks in villages and download it to the Internet when passing by an Access Point in a remote town, or alternatively to an Intranet even in town (e.g., at local government offices), or anywhere needed. Similarly MAPs may download some requested information from the Internet and bring it to the villages, e.g., educational materials. DakNet has the potential to support Internet/Intranet messaging (e.g., email and audio/video messaging, mobile e-commerce), distribution of information (e.g., public health

announcements, community bulletin boards, news, and music), and collection of information (e.g., environmental sensor information, voting, health records, and census).

## 5. Concluding remarks and Future trends

At the top level of our taxonomy, we have divided routing techniques for opportunistic networks between algorithms that do exploit some form of infrastructure, and algorithms that do not. Actually, the vast majority of papers in the literature follow this distinction. We believe that a very interesting area still to be investigated is how to design multi-tier opportunistic networks. In this sense, the data MULEs and message ferrying architectures are the most promising approaches. They are susceptible to various improvements but have the potential to be utilised as bases for more general and global architectures. For example, in the data MULEs approach, lower-level nodes exploit the higher level and more capable mobile devices (the MULEs), which, in turn, exploit a further infrastructure level, i.e., the Access Points. However, routing algorithms exploited at each layer are pretty trivial or do not exist at all. Instead, we can envision a multi-tier *fully opportunistic* network. In such a network, each level of the infrastructure is an opportunistic network in which nodes may exploit routing algorithms to communicate among themselves, and may rely on the upper levels of the infrastructure to reach nodes that are too far away. For example, a low level can consist of devices such as PDAs, or smart phones. An opportunistic routing algorithm can make those devices able to communicate with each other. To reach nodes too far away for such routing to be effective, a higher level consisting, for example, of a "city-bus network" might be used. In this scenario, buses will act similarly to MULEs. However, multi-hopping will be used also at this level of the network via a (possibly different) opportunistic routing algorithm. This will enable connection among different clouds of the lower-tier devices just by relying on the city-bus network. Clearly, the city-bus network might exploit further infrastructure levels such as a mesh network formed by Access Points, or even access the Internet through standard Wi-Fi Access Points. Designing such an opportunistic multi-tier network

is one of the most interesting challenges that can currently be envisaged. Once designed and developed, such a network might actually represent a fundamental building block for the Next-Generation Internet.

# V. Encoding Techniques for Reliable Data Transmission in Opportunistic Networks

## 1. Introduction

The diffusion of pervasive, sensor-rich, network-interconnected devices embedded in the environment is rapidly changing the Internet. The nature of pervasive devices makes wireless networks the easiest solution for their interconnection. Furthermore, in a pervasive computing environment, the infrastructure-based wireless communication model is often not adequate: it takes time to set up the infrastructure network, while the costs associated with installing infrastructure can be quite high [Con04]. Therefore multi-hop ad hoc wireless technologies represent one of the most promising directions to further extend the current Internet, and diffuse traditional networking services even more widely ([CCL03] [MC04]). The heterogeneity of devices to be interconnected (from small sensors and actuators to multimedia PDAs), and the large spectrum of communication requirements (from few meters coverage and few kilobits bandwidth to city-wide coverage and broadband communications), have produced a set of multi-hop ad hoc network technologies. On the one hand, we have sensor networks for communications among small sized, low cost, and low energy-consuming devices (sensors) for which a high data rate is not necessary [AVAS06]. On the other hand, we have mesh networks that, by exploiting an infrastructure of wireless mesh routers, guarantee the exchange of multimedia information among devices located inside an urban area [BCG05]. The work in [Conar] provides a complete overview of the ad hoc networking techniques by presenting their principles and application scenarios, and pointing out the open research issues. Among the ad hoc networking techniques,

*opportunistic networks* [PPC06b] represent the most interesting scenario for the application of the encoding techniques analyzed in this chapter. Opportunistic networks represent the evolution of the multi-hop ad hoc network concept in which the end-to-end connectivity constraint is released. Indeed, typically, mobile ad hoc networks rely on the end-to-end principle, i.e. a path must continuously exist between the sender and the receiver for successful communications. In reality, end-to-end connectivity is a strong requirement only for interactive services such as VoIP, gaming, video streaming; many other applications can still correctly operate relaxing the end-to-end constraint, e.g., data applications like messaging, e-mails, data sharing, etc. In principle they can operate even if a sender-receiver path never exists [PPC06b]. In opportunistic networks, a node stores the messages in its local memory until a suitable forwarding opportunity exists. In this way packets are not discarded during network disconnections but are locally stored. The communication is still multi-hop with intermediate nodes acting as routers that forward the messages addressed to other nodes but, in this case, forwarding is not "on-the-fly" since intermediate nodes store the messages when no forwarding opportunity exists (e.g., there are no other nodes in the transmission range, or neighbouring nodes are considered not useful to reach the destination), and exploit any contact opportunity with other mobile devices to forward information.

In all these scenarios, a big challenge is efficient data distribution. Applications like messaging, content distribution and peer-to-peer applications are becoming more and more widespread in the today Internet. Thanks to their decentralized features, they are likely to play a major role in ad hoc environments such as opportunistic networks. One of their major requirements are networking techniques to efficiently convey data to possibly large sets of users. In the legacy wired Internet, researchers have proposed solutions based on IP multicast and, more successfully, on peer-to-peer overlay networks. While p2p solutions for wired networks are quite well established, designing similar systems to enable efficient data distribution over mesh, opportunistic and delay-tolerant networks is a big challenge still far to be satisfactorily addressed. In the field of sensor networks, content-distribution and p2p applications are not very likely. However, typical sensor network applications such as environmental monitoring require efficient data distribution techniques, too. The very characteristics of wireless communications make

legacy solutions for data distribution designed for the wired Internet not effective in ad hoc environments, even in static scenarios. Systems designed for ad hoc networks cannot assume that "bandwidth is for free", as most legacy systems do. In addition, they have to efficiently cope with sudden changes of the wireless links characteristics. Dynamic topology reconfigurations, due either to user mobility or to energy management techniques that temporarily switch off some nodes, add further dimensions to the problem. These constraints are peculiar of wireless multi-hop ad hoc networks, and legacy systems are typically not able to cope with them. Data distribution systems, either targeted to p2p and content distribution paradigms, or designed for sensor network applications, are thus today an exciting research area for multi-hop ad hoc networking environments. Encoding techniques are an important building block to address many issues that arise in these systems. The main idea of encoding techniques applied to networking protocols is not to send plain data, but to combine (encode) blocks of data together, in such a way that coded blocks can be interchangeable at receivers. In the simplest example, a source node willing to send $k$ packets actually encodes the $k$ packets into n encoded packets, with $n \gg k$. Encoding is performed so that a receiving node has not to exactly receive the $k$ original packets. Instead, any set of $k$ packets out of the $n$ encoded packets generated at the source is sufficient to decode the $k$ original packets. Different receivers might get different sets of encoded packets, and still be able to decode the same original information. These encoding algorithms, known as *erasure coding*, have been originally applied to implement efficient reliable multicast protocols for wired Internet [RV98]. Recently they have found interesting application scenarios in wireless ad hoc networks. For example, erasure code techniques have been used in [KFC04] for the design of a reliable point-to-point protocol for data transmission over wireless sensor networks. Other applications of erasure code techniques over wireless sensor networks are reported in [KOX05] and [DPR05]. In [KOX05] they have been exploited as a means to achieve minimum-energy data transmissions, while in [DPR05] they have been used to support distributed data caching over the network and facilitate subsequent data retrieval. Erasure codes have also been used to support speech communications over well-connected mobile ad hoc networks [DCG+03] since they are able to reduce the total transfer delay experienced by data packets,

and make it adequate to real-time applications. In the case of multi-hop ad hoc networks erasure code techniques have been generalized by introducing several encoding phases of the same data. This generalization is known as *network coding*. With *network coding*, data packets are encoded both at source nodes, and also at intermediate hops in the path towards receivers. Assuming that the communication paradigm is one-to-many, the source node encodes locally generated data packets and sends them to a subset of current neighbors. Such intermediate nodes generate a new set of encoded data packets based on the received packets, and further disseminate them towards the final destinations. Network coding has shown to be a very promising solution for data dissemination in multi-hop ad hoc networks [DEH+05]. Actually, it is able to provide very high reliability and exploit bandwidth very efficiently. It has been successfully applied to achieve optimal energy consumption in multicast [WCK05b], unicast [WCK05a], and also broadcast [WFLB05] transmissions over mobile ad hoc networks. A lot of attention has also been devoted to possible applications of network coding over mesh networks [KRH+06] [HBT06], as well as over opportunistic [WLB05], and vehicular ad hoc networks [LPY+06]. In this chapter we provide a detailed overview of the research efforts on encoding techniques for multi-hop ad hoc networks. In Sections 2 and 3 we deeply describe the basic encoding techniques network coding is based on. Section 4 deals specifically with network coding, describing its operations and discussing its applications to multi-hop ad hoc networks. Finally, Section 5 draws conclusions and highlights open issues of this field.

## 2. Coding Theory: Motivation and Basic Concepts

Reliable transmission of data over *noisy channels* has been a major concern to the communication engineering for a long time and appropriate control systems as well as recovery mechanisms for corrupted data have been thoroughly studied. A real breakthrough in this field was achieved in 1948 when Claude Shannon [Sha48] demonstrated that by efficiently *coding* messages at the sender before transmission and conversely decoding (possibly corrupted) messages that arrive at the receiver, it is possible to repair the effects of a noisy channel. The decoding

system acts on the corrupted messages and strives to reconstruct the original messages. Encoding of data is equivalent to adding some *redundancy* to transmission so as the encoded data which is transmitted, i.e., the *code*, stores the information that has originally been produced at the source more robustly. Indeed, redundancy is exploited at the receiver both to realize that the incoming code is corrupted and to correct it. In the first case the code is said to be *error detection code*, in the latter, *error correcting code*. The principle behind error detecting and correcting codes is also embedded in everyday language. Each language, in fact, is characterized by a vocabulary composed by a certain number of words. Words are sequences of letters (symbols) belonging to a particular alphabet. Albeit the total number of words included in a vocabulary is very high, it is far less than the total number of possible combinations of the alphabet letters. Thank to this property, if a misprint occurs in a long word, it can easily be recognized because the word (likely) changes into something that does not correspond to any other word of the vocabulary and rather resembles the correct word more than it resembles any other known word. Adding redundancy to the information to transmit is equivalent to *mapping* the set of all the possible pieces of information that a source can generate, hereafter referred to as words, into a far bigger information-set of so-called *code-words*. The mapping function is such that each word has a correspondent code-word associated to it whereas a code-word may not correspond to any significant word produced at the source. The mapping process is referred to as *encoding* process. At the destination an inverse mapping is performed named *decoding* to reconstruct the original word transmitted by the source. Fig. V.1 shows the encoding, transmitting and decoding processes for the data originated at a source. The decoding algorithms adopted at the receiver can be of different types. A *complete* decoding algorithm decodes every possibly received code-word (even corrupted) into a corresponding word. However, in some situations an *incomplete* decoding algorithm could be preferable, namely when a decoding error is very undesirable. When receiving new signals from the channel a new code-word should be read. The receiver infers each single symbol (bit) of the code-word from the signals received from the channel, so, for each received signal it has to decide whether it is much more similar to a *0*-signal or to a *1*-signal. Luckily, in most cases this decision is straightforward. Otherwise, it might be preferable to put a "?" instead

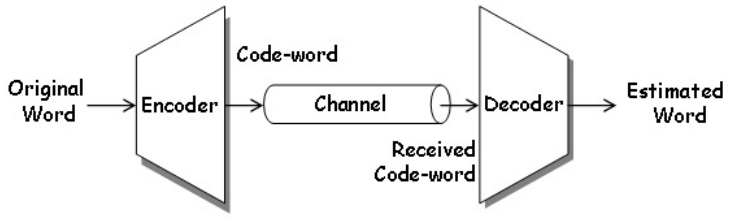


Figure V.1.: Encoding, transmitting and decoding processes for the data originated at a source.

of guessing whether the symbol is 0 or 1. In the subsequent decoding stage the receiver has to reconstruct the original word from the received code-word possibly including some "?"-s. An incomplete decoding algorithm *corrects* only those code-words which contain few errors whereas those with more errors cause decoding failures and give rise to so-called *erasures* in the sequence of the received words. The receiver can either ignore erasures or, if possible, ask for retransmission. The success of an encoding/decoding system is strictly related to the characteristics of the channel over which transmissions occur and to the amount of redundancy which is introduced to transmission. Shannon formulated this relation in terms of *information rate* and *channel capacity*. He defined the *information rate* as the ratio between the number of significant words of a vocabulary and the total number of possible code-words that can be represented given a certain alphabet of symbols[1]. In addition, he defined the *channel capacity* as the total amount of information that a channel can transmit. Finally he demonstrated that, in case the information rate is lower than the channel capacity, there exist an efficient system for encoding and decoding such that the probability of corruption due to transmission is lower than $\varepsilon$ , for each arbitrarily small $\varepsilon > 0$ . This means that it is possible to transmit any information with an arbitrarily small probability of error by using the right encoding technique. The code system proposed by Shannon, known as *Shannon*

[1] Given *code* $C$ the set of code-words corresponding to a certain vocabulary and given $n$ the number of bits over which code-words are represented, then the information rate (or just the *rate*) of the code $C$ is defined as $\frac{\log_2 |C|}{n}$, where $|C|$ gives the total number of code-words of the set $C$. Please, note that code $C$ is only a sub-set of all the elements that can be represented over $n$ bits and that each element of it matches a word of the vocabulary. Hence the cardinality of code $C$ is equal to the cardinality of the vocabulary.

*code*, relies on a *probabilistic* approach and uses a random generation for code-words to assign to the words produced at the source. This choice is motivated by the fact that it is quite unlikely that a random process generates code-words similar to one another. The *similarity* between code-words is to be intended in terms of the *Hamming distance* that gives the total number of symbols that two code-words are differing in[2]. If for any couple of code-words generated by a coding system the code-words are sufficiently far from each other (differ in many symbols) then they are unlikely to be confused. Hence, whenever a code-word is sent over a channel and modified by noise, it is quite improbable that the corrupted code-word arriving to destination can be confused with another code-word. This would happen if only had the noise reduced the Hamming distance between the two code-words so much that they could be considered the same. If the initial Hamming distance is quite long, then this should not happen and the reconstruction of the original source word should be easy and unique. In the Shannon coding system a source word is a string of $m$ symbols ($m > 0$) generated at the source site. These strings have to be encoded into strings of size $n$ symbols i.e., code-words, with $n$ higher than $m$, and subsequently sent. Choosing $n$ higher than $m$, words generated at the source are mapped over a bigger space with a higher dimension (so the Hamming distance between code-words is likely to be high). The choice of the correct lengths, $m$ and $n$, of both words and code-words can be made once targeted a specific $\varepsilon > 0$. Then an appropriate *encoding/decoding system* can be selected such that the probability not to reconstruct the original word sent by the source is lower than $\varepsilon$. Unfortunately Shannon code cannot be implemented. Specifically, the decoding system cannot be implemented because it should evaluate the Hamming-vicinity of the received corrupted code-word to any other code-word of the code and select the code-word with the minimum distance as the un-corrupted code-word (this search has a dramatic cost) and then apply the inverse mapping to reconstruct the original word. Many codes have been conceived after the original Shannon code. They have such algebraic characteristics that not only can the decoding process be implemented but it can also be very fast. Anyway, the *delay* caused by both encoding and decoding computations as well as the increase in

[2] If $\underline{x} = (x_0,\ x_1,\ \ldots,\ x_{n-1})$ and $\underline{y} = (y_0,\ y_1,\ \ldots,\ y_{n-1})$ are two tuples, each one long n bits, then their Hamming distance is defined as follows. $d(\underline{x},\ \underline{y}) = |\{\ i \mid 0 \leq i < n,\ x_i \neq y_i\}|$, where $|.|$ is the cardinality of the set ".".

the total transmission time due to the lengthening of the data to transmit, has been the main limiting factor to the diffusion of encoding techniques, and still remains their main drawback. Error correcting codes have mainly been used, so far, in dedicated and critical communications systems (e.g., satellite systems) where they serve as a means to avoid expensive and long-lasting retransmissions. In these systems, encoding and decoding algorithms are typically implemented on dedicated hardware which allows overcoming the computational burden. Error correcting codes are also used in other special cases e.g., in modems, over wireless or otherwise noisy links, in order to make the residual error rate comparable to that of dedicated, wired connections. Error detection codes have mainly been used, instead, in the context of general purpose computer communications to discard the corrupted frames arriving from the communication channel, actually, they are quite easier to implement than error correcting codes. Error detection codes (see for example the Cyclic Redundancy Checksums (CRCs)) are managed at the lowest layers of the protocol stack (physical and data-link layers) that are typically HW-implemented in the network adaptor. While error correcting codes within streams of bits have seldom been implemented in these general purpose systems, especially in SW, correction of errors can be implemented quite more efficiently if focusing on missing packets rather than on corrupted symbols (i.e., bits). In fact, SW-implemented codes are actually being successfully developed in the last few years. They are known as *Erasure Codes* because they deal with erasures, i.e. missing packets in a stream. Erasure codes are implemented above the data-link layer where information is organized in *packets* rather than bit-streams (frames) and the channel noiseness is only perceived, after error processing and *detection* of the lower-layer protocols, through packet loss i.e., erasures. Erasures originate from errors that cannot be corrected at the data-link layer (but those are not frequent with properly designed and working hardware), or, more frequently, from congestion in the network which causes otherwise valid packets to be dropped due to lack of buffer space. Erasures are easier to deal with than errors since the exact position of the missing packets is known. Recently many efforts have been spent in the construction of *Erasure Codes*. They rely on the same concepts of the aforementioned error detecting and correcting codes but operate on packet-sized data objects rather than on bit-streams, and can be imple-

mented in software using general purpose processors. The motivation for recent deployment of high-layer erasure codes is slightly different from that of previous low-layer error codes. Erasure codes are attracting interest in the new emerging communication scenarios where existing protocols are becoming unsuitable, or at least inefficient, because for example, are based on frequent handshaking between the communicating peers, or because assume the existence of a continuous end-to-end connection between peers. Erasure codes have the potential to add to communication protocols i) reliability of transmissions, ii) scalability to the increasing number of hosts, iii) adaptability to the topology changes of a network (e.g., due to mobility or even power saving strategies), and iv) robustness to node/link failures. These are very promising features especially in scenarios like multicast transmissions, wireless networks (e.g., ad hoc, sensor, vehicular ad hoc, etc.), satellite networks, delay tolerant networks, and intermittently connected networks. Software implementation for erasure codes is surely a challenging task due to the computational complexity. However, it has also been demonstrated that it can efficiently be done [Riz97a] [Riz97b] [RV98]. The rest of the chapter will focus on erasure codes and their more recent extension, network coding. In the following section some types of erasure codes will be presented, from the oldest to the most recent ones and some examples of applicative scenarios will be given for each of them, as well.

## 3. Erasure Codes

Erasure codes are generally considered implementations of so-called *Digital Fountains*. Digital Fountains are ideal, unlimited data-streams injected into the network by one or more source nodes. Nodes that are willing to receive information from those source nodes need to catch data from the stream until they have collected enough information to fulfil their needs. The way receivers catch data from the unlimited stream resembles the way people quench their thirst at a fountain. No matter which drops of water fall down on earth and which are drunk, the only important thing is that enough drops are caught. Source nodes generate digital fountains sending out the data they are willing to send and adding redundancy to

it via encoding. The redundancy added is practically unlimited because the generated data flow is infinite. Namely, suppose that $k$ data packets are originated at a source node. After encoding an unlimited stream of data packets is generated and injected into the network. The receiver node needs to receive *exactly* $k$ data packets to be able to reconstruct the original source data. Moreover, *any* $k$ data packets are suitable to the scope. Digital Fountains have been introduced in [BLM02] and so far have mainly been used in conjunction with multicast protocols to improve reliability of transmissions. Multicast protocols suffer from the so-called feedback implosion that occurs when multiple receivers ask for retransmission of data to the server. Multiple retransmissions are due to the inevitably different loss patterns that affect different receiver nodes. Clearly, it is not affordable to sending nodes to manage multiple retransmissions of different data packets towards different receivers. If the server generates a digital fountain instead, receiver nodes only have to receive an amount of data packets that corresponds to the exact number of original data packets. Indeed, *any* set of packets of size equal to the number of original data packets is appropriate. Hence, whenever receivers miss some packets it is sufficient that they wait for the subsequent redundant packets to come. No retransmissions are needed towards different receivers, no feedback channel is strictly necessary. Content dissemination in environments with a vast number of receivers having heterogeneous characteristics can greatly benefit from this approach (see, for example, distribution of software, archived video, financial info, music, games, etc.). In these environments receivers will wish to access data at times of their choosing, they all will have their own access speeds and, in addition, their access times will probably overlap with one another. Besides addressing the retransmissions' issue, in this case digital fountains allow receivers to join the transmitted stream at any point and without requiring any synchronization among them. Synchronization is neither needed between senders and receivers. Other challenging target scenarios for digital fountains are satellite networks where channel characteristics, i.e., high latency and limited as well as intermittent capacity, make retransmission-based solutions totally unworkable. Finally, wireless networks are emerging as a very promising scenario for digital fountains because of the hardly unreliable communications and the common asymmetry between upstream and downstream channels. In addition, the recent deploy-

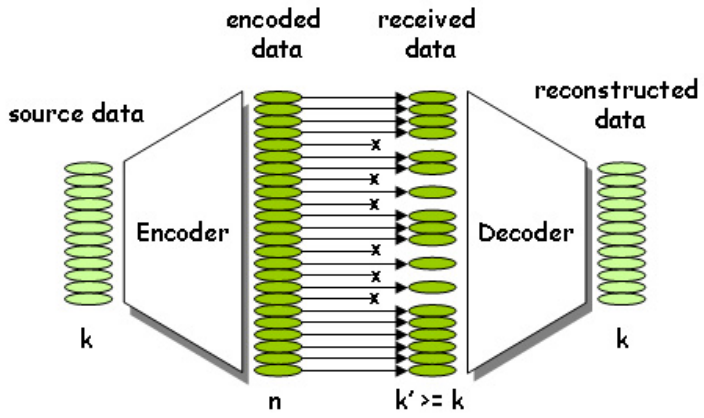


Figure V.2.: A graphical representation of the encoding and decoding processes of an *(n, k)*-code.

ment of opportunistic communication paradigms in the framework of the wireless networks characterized by intermittent connectivity looks quite an interesting application field, as well. Although very promising, it should be noted that digital fountains are only an abstract concept. An unlimited stream of data is not feasible in practise, neither advisable, because it has the potential to flood and congest the network. During the last few years multiple approximations of digital fountains have been proposed trying to find a trade-off between efficacy and computational costs (see [RS60]-[SLL06]). All these approximations can be expressed in terms of *(n, k)-codes*, where $k$ gives the number of source data blocks[3] which are encoded while $n$ gives the number of data blocks that the encoder produces from the original $k$ blocks, and that are actually sent over the network. Fig. V.2 shows the encoding and decoding processes of an *(n, k)*-code. During transmission over the network, some encoded data blocks may get lost due to congestion or corruption and only a sub-set of encoded blocks of size $k'$, $k \leq k' \leq n$ , arrives to destination and enters the decoder. The decoder gives back the original $k$ source data blocks. As will be shown in detail in the next sections, to be able to reconstruct the source

[3] Here *block* is a generic piece of data. In the following sections *block* will be used to refer either a symbol or a packet or even a set of packets. Specific interpretations will be given when needed.

data blocks, the decoder needs at least $k$ blocks of encoded data. Some implementations actually need to get at least $more$ than $k$ blocks to decode the original data. In the optimal case, an *(n, k)*-code allows the receiver to recover from up to $n - k$ losses in a group of $n$ encoded blocks. In the remainder of this section insights on four types of erasure codes will be given: Reed-Solomon codes, Tornado codes, Luby-Transform codes, and Raptor codes. The order followed in the presentation is chronological since these codes have been conceived in a temporal sequence each one being an improvement of the previous one.

## 3.1. Reed-Solomon Codes

Reed-Solomon codes have been introduced in [RS60] as implementation of digital fountains. They are currently used to correct errors in many systems including wireless or mobile communications (e.g., cellular telephones, microwave links, etc.), satellite communications, digital television, and high-speed modems such as ADSL and xDSL. They are also used in storage devices (e.g., tape, Compact Disk, DVD, barcodes, etc.). Assume a source data block is a word and let a sequence of $k$ words be represented by a vector, say $\underline{x}$, of $k$ elements. Encoding is represented by an encoding function $f(.)$ which is applied to $\underline{x}$ and produces an encoded vector of $n$ code-words. When the encoding function is *linear*, the code is said to be linear too. Reed-Solomon codes are a special case of linear codes as will be better explained later, after the following brief introduction to general linear codes.

### 3.1.1. Linear Codes

Linear codes can be represented by a matrix $G$ (encoding matrix) and encoding can be represented by a matrix-vector multiplication. Hence, a vector of code-words $\underline{y}$ corresponding to a vector of words $\underline{x}$ is simply given by the product $G \cdot \underline{x}$. Remembering the definition of an *(n, k)*-code, the encoding process looks

as follows.

$$\underline{y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_{n-1} \end{pmatrix} = G \cdot \underline{x} = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & \cdots & g_{0,k-1} \\ g_{1,0} & g_{1,1} & \cdots & \cdots & g_{1,k-1} \\ g_{2,0} & g_{2,1} & \cdots & \cdots & g_{2,k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n-1,0} & g_{n-1,1} & \cdots & \cdots & g_{n-1,k-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_{k-1} \end{pmatrix} \tag{V.1}$$

where $\underline{x} = (x_0,\ x_1,\ \ldots,\ x_{k-1})^T$ is the vector of $k$ source words, $\underline{y} = (y_0,\ y_1,\ \ldots,\ y_{n-1})^T$ is the vector of $n$ code-words, and $G_{(n \times k)}$ is the encoding matrix. The encoding matrix must have rank $k$. Suppose the destination node receives at least $k$ (otherwise decoding is not possible) out of the $n$ code-words produced by the encoder at the source, and let $\underline{y}'$ be a vector of $k$ elements picked up among the received code-words. The encoding process that has generated this vector follows.

$$\underline{y}' = \begin{pmatrix} y_{i,0} \\ y_{j,1} \\ \vdots \\ \vdots \\ y_{l,k-1} \end{pmatrix} = G' \cdot \underline{x} = \begin{pmatrix} g_{i,0} & g_{i,1} & \cdots & \cdots & g_{i,k-1} \\ g_{j,0} & g_{j,1} & \cdots & \cdots & g_{j,k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{l,0} & g_{l,1} & \cdots & \cdots & g_{l,k-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_{k-1} \end{pmatrix} \tag{V.2}$$

The encoding matrix $G'_{(k \times k)}$ is a $k \times k$ matrix obtained by extracting from $G_{(n \times k)}$ those rows that correspond to the elements of the vector $\underline{y}'$. So, for example, if the *j*-th code-word of the original code-word vector (i.e., $y_j$) is inserted as the second element in the vector $\underline{y}'$ (i.e., $y_{j,1}$), then the *j*-th row of the matrix $G_{(n \times k)}$ is picked up and inserted as the second row in the matrix $G'_{(k \times k)}$. Clearly, decoding means finding the solution to the linear equation $G' \cdot \underline{x} = \underline{y}'$, as follows.

$$\underline{x} = G'^{-1} \cdot \underline{y}' \tag{V.3}$$

Note that the destination must be sure to identify the row in $G_{(n \times k)}$ corresponding

to any received element of $\underline{y}$. Note also that the set of rows corresponding to $\underline{y}'$ have to be linearly independent.

### 3.1.2. Encoding process of RS-codes

As has been introduced above, Reed-Solomon codes are a special case of linear codes. Source words are seen as coefficients of a polynomial of degree $k-1$, whereas code-words are seen as values of the polynomial worked out at $n$ different points that can be chosen arbitrarily. Let the polynomial be as follows.

$$p(x) = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + \ldots + a_{k-1} x^{k-1} \tag{V.4}$$

where $a_0,\ a_1,\ \ldots,\ a_{k-1}$ are the $k$ words generated at the source for transmission and $p(x)$ is a single code-word obtained by evaluating the polynomial at the point $x$. The encoding process for a Reed-Solomon *(n, k)*-code is thus as follows.

$$\begin{pmatrix} p(x_0) \\ p(x_1) \\ p(x_2) \\ \vdots \\ \vdots \\ \vdots \\ p(x_{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 & \ldots & x_0^{k-1} \\ 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \ldots & x_2^{k-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & x_{n-1}^3 & \ldots & x_{n-1}^{k-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_{k-1} \end{pmatrix} \tag{V.5}$$

where $x_0, x_1, \ldots, x_{n-1}$ are the $n$ points selected for evaluation of the polynomial. They can be chosen arbitrarily, for example, for simplicity of encoding, or alternatively they can be *all* the possible integer values that can be represented over the number of bits available. The encoding matrix of Reed-Solomon codes has a special form characterised by a geometric progression in each row. Such matrices are named *Vandermonde* matrices.

### 3.1.3. Decoding process of RS-codes

The decoding process consists in reconstructing all the polynomial coefficients $a_0,\ a_1,\ \ldots,\ a_{k-1}$ in a unique way. As is commonly known, for this to be possible, it is sufficient to know the value of the polynomial at exactly $k$ points, i.e., receiving $k$ code-words is sufficient. Hence, assuming that the identity (e.g., the sequence number) of the code-words that have arrived to destination is known, the coefficients of the polynomial can be derived at the destination site by solving the following system of equations.

$$\begin{pmatrix} y_{i,0} \\ y_{j,1} \\ \vdots \\ \vdots \\ y_{l,k-1} \end{pmatrix} = \begin{pmatrix} 1 & x_{i,0} & x_{i,0}^2 & \ldots & x_{i,0}^{k-1} \\ 1 & x_{j,1} & x_{j,1}^2 & \ldots & x_{j,1}^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{l,k-1} & x_{l,k-1}^2 & \ldots & x_{l,k-1}^{k-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_{k-1} \end{pmatrix} \tag{V.6}$$

The matrix utilized in the decoding process is a sub-matrix of the encoding matrix and is obtained by selecting the $k$ rows which correspond to the arrived code-words (in the example the *i*-th, *j*-th, ..., and the *l*-th rows have been selected). The system admits a solution if the matrix is non singular. The determinant of the above $k \times k$ Vandermonde matrix corresponds to the following expression.

$$det(V) = \prod_{0 \le l < t < k} (\hat{x}_t - \hat{x}_l) \tag{V.7}$$

given $\underline{\hat{x}} = (\hat{x}_0,\ \hat{x}_1,\ \ldots,\ \hat{x}_{k-1})^T = (x_{i,0},\ x_{j,1},\ \ldots,\ x_{l,k-1})^T$ the second column of the Vandermonde matrix. Hence, the determinant is non-null if and only if all the $\hat{x}_i$-s are non-null and different from each other. It should finally be noted that, to allow decoding Reed-Solomon codes, the encoding matrix has to be *known* at *both* the source and destination sites.

### 3.1.4. Arithmetic of RS-codes

Reed-Solomon codes are based on a mathematics area known as *Galois fields* or *finite fields*. A Galois field $GF(p)$, also called *prime field*, is a field that includes $p$ elements, from $0$ to $p-1$, with $p$ prime. It is *closed* under additions and multiplications modulo $p$. Operating on a prime field is relatively simple, since field elements can be thought of as integers modulo $p$, and sums and products are just ordinary sums and products, with results computed modulo $p$. Galois Fields can be used for the representations of the elements involved in RS-codes, i.e., the blocks of source data, the blocks of encoded data, and the elements of the encoding matrix. However, from the implementation standpoint, this is not feasible or at least efficient for two main reasons. Firstly, the number of bits necessary to represent an element of a Galois Field is $\lceil log_2 p \rceil > log_2 p$. This introduces inefficiency for encoding and also for computation of operations because the operand sizes may not match the word size of the processor. In addition, operations modulo $p$ are expensive because they need a division to be applied. It is much more efficient to work on a so-called *Extension Field* $GF(p^r)$ that includes $q = p^r$ elements, with $p$ prime and $r > 1$. Field elements can be represented as polynomials of degree $r-1$ with coefficients in $GF(p)$. Operations in Extension Fields with $p = 2$ can be extremely fast and simple to implement. Modulo operations are not needed. Efficient implementations can be realized exploiting only XORs and table lookups. A thorough dissertation on the arithmetic of Extension Field can be found, for example, in [Lin82] and [Lin70].

### 3.1.5. Systematic Codes

When the code-words include a verbatim copy of the source words, the code is said to be *systematic*. This corresponds to including the identity matrix $I_k$ in the encoding matrix. The advantage of a systematic code is that it simplifies reconstruction of the source words in case very few losses are expected. In Fig. V.3 for example, two out of the $k$ code-words arrived at destination are actual original words. Hence, the system of equations that must be solved to reconstruct the original words includes $k-2$ equations instead of $k$. By using $x_i \in GF(q = p^r)$,

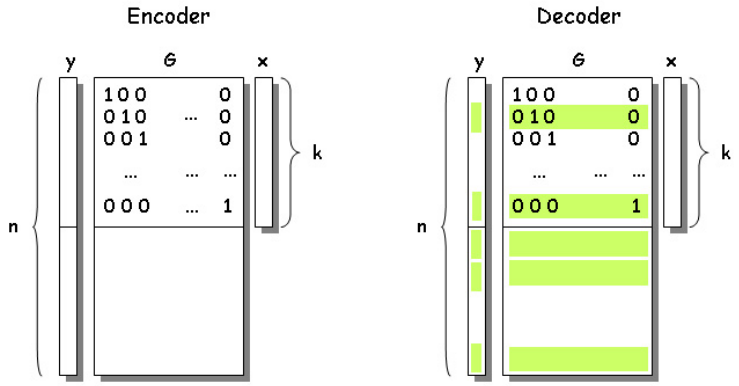


Figure V.3.: The encoding/decoding process in matrix form, for systematic code (the top $k$ rows of $G$ constitute the identity matrix $I_k$). $y'$ and $G'$ correspond to the green areas of the vector and matrix on the right.

it is possible to construct encoding matrices with up to $q-1$ rows corresponding to the total number of non-null $x_i$'s of the field. The number of columns of the encoding matrix is instead $k$ corresponding, as usual, to the total number of original source words and the relation $q-1 \geq k$ must hold true. As a result, the maximum size of the encoding matrix is $(q-1) \times (q-1)$. The encoding matrix can further be extended with the identity matrix $I_k$ to generate systematic codes that help simplify the decoding process. Hence, the final maximum size of the encoding matrix is as follows.

$$((q-1)+k_{max}) \times k_{max} = ((q-1)+(q-1)) \times (q-1) = 2(q-1) \times (q-1) \quad \text{(V.8)}$$

where $k_{max}$ is the maximum number of source words that can be encoded over $GF(q)$.

### 3.1.6. Applications

A major hurdle in implementing a digital fountain through standard Reed-Solomon codes (RS codes) is the unacceptably high computational time caused by both en-

coding and decoding. It is of order $O(n^2)$, $n$ corresponding to the number of generated code-words. The large decoding time for RS codes arises from the *dense* system of linear equations which is generated. However, efficient implementations of RS codes are actually feasible as has been shown in [Riz97a] and [Riz97b]. By means of thorough optimizations the developed code performs encoding and decoding on common microprocessors at speeds up to several MB/s. The code has been tested on a wide range of systems, from high end workstations to old machines and small portable systems (see e.g., DECstation 2100, SUN IPX, Sun Ultra1, PC/FreeBSD) with CPU speeds ranging from 8 up to 255 MHz. This variety is motivated by the fact that erasure codes can be used in very different contexts and with very different speed requirements. Results effectively demonstrate the feasibility of *software FEC (Forward Error Correction)* and advocate its use in practical, real applications. The same code has finally been utilized to implement a reliable multicast data distribution protocol [RV98]. Namely, the use case referred is a large file transfer from a server to multiple receivers. The file is already encoded (off-line) and stored at the server ready for transmission. Receivers can join the multicast session whenever they need, generally at different time instants. The server needs a counter of the maximum number of encoded packets that have to be sent out to each receiver. This maximum number is the sum of the number of packets containing original data, and the number of packets containing redundant data. Each time a new receiver joins the session the server simply re-initializes the counter to the maximum value. Then, it continues transmitting the file from exactly the same point it had arrived when the last receiver joined the session and in the same order already begun. The counter is decremented after sending a packet. Once arrived to the end of the file, the server loops through the file until all the requests are fulfilled, i.e., until the counter goes to zero meaning that all the receivers have received the appropriate number of packets. Since with RS-codes reconstruction of source data is possible after having successfully received any $k$-set of distinct encoded packets ($k$ being the size in packets of the original data set), each receiver leaves the session when it has received enough packets to decode the entire file. When a new receiver joins the session, transmission of data does not need to start again from the beginning of the file. The new receiver receives the same packets the other receivers are re-

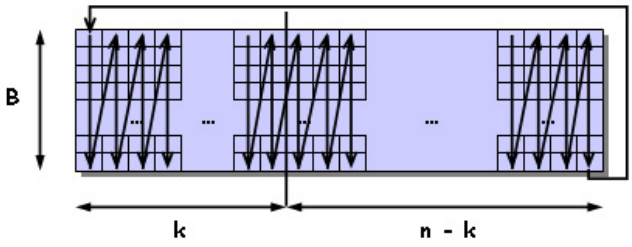


Figure V.4.: The arrangement of data and the transmission order at the server.

ceiving, but it will stop receiving later than the others. Fig. V.4 shows the scheme followed by the server to transmit the file content. The source file is sub-divided into $B$ blocks of data (on the vertical axis of the figure). The size of each block is such that it can be sent over $k$ packets, thus it can be said that each block includes $k$ packets. Each block is encoded separately, so redundancy is added to each single block independently from the others. Hence, assuming a systematic code, each encoded block includes $k$ original packets plus $n - k$ redundant packets (on the horizontal axis of the figure). Finally, the server sends out packets by repeatedly looping through all the blocks. Namely, packets scheduled for transmission are picked from consecutive blocks, rather than sequentially from the same block. Such *interleaving* protects single blocks from the effects of a burst of losses since it is preferable that losses are spread among all the blocks rather than concentrated in a unique block. Encoding is applied to single blocks of data rather than to the entire file to reduce the computational burden of both encoding and decoding. As has been mentioned above, in fact, the encoding and decoding costs are quadratic in the size of the encoded block ($O(n^2)$). Therefore, given that $n > k$, if $k$ were the size of the entire file, the computational cost would be very high, whereas, by encoding over smaller chunks of file at a time is much more efficient (even though encoding must be repeated more times, once for each block). The transmission scheme of Fig. V.4 is known as *data carousel* (see also [BLM02]). It limits both encoding and decoding times, allows multiple receivers to be served at the same time, and eliminates the need for several retransmissions towards different receivers. However, it has a drawback in the *receipt overhead*. In

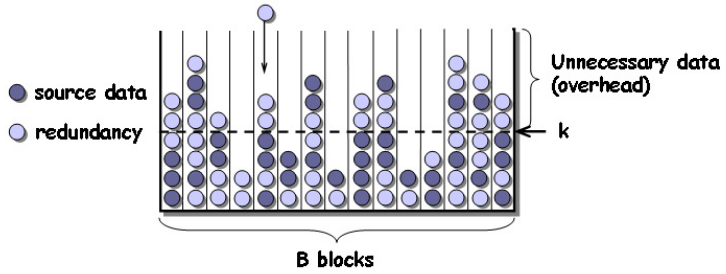


Figure V.5.: Waiting for the last blocks to fill.

fact, since transmission loops through different blocks, after having completely received and decoded a certain number of blocks, a receiver can potentially receive and discard packets belonging to the blocks already decoded before receiving the missing packets necessary to decode the last blocks. The receipt overhead is measured in terms of unnecessary received packets (see Fig. V.5). Starting from the achievements of [Riz97b], [Riz97a], and [RV98], an implementation of RS-Codes has been provided also for wireless sensor networks. Authors of [KFC04] propose a point-to-point data transmission protocol for WSNs that *combines* multiple strategies, namely i) erasure codes, ii) re-transmission of lost packets and iii) a route-fix technique. RS codes are used to add redundancy to transmission such that the receiver is able to reconstruct original data despite losses or congestion (up to a certain point). Nevertheless, as underlined in [KFC04], re-transmissions cannot be eliminated completely. Rather it is necessary to provide integration between encoding and re-transmission of data and to fine-tune this integration for better efficiency. Re-transmissions are managed hop-by-hop such that the point of retransmission is moved progressively forward towards the destination node. Experimental results show that these two strategies together are very efficient and result in good reliability. Nevertheless, it has also been noted that both link-level re-transmissions and erasure codes have poor performance in case of route failures. When a route is no more available, consecutive losses occur and re-transmissions are of no use because the route towards the destination does not exist any more. Neither erasure codes are helpful in this case because receivers cannot receive the minimum amount of encoded messages that is needed to re-

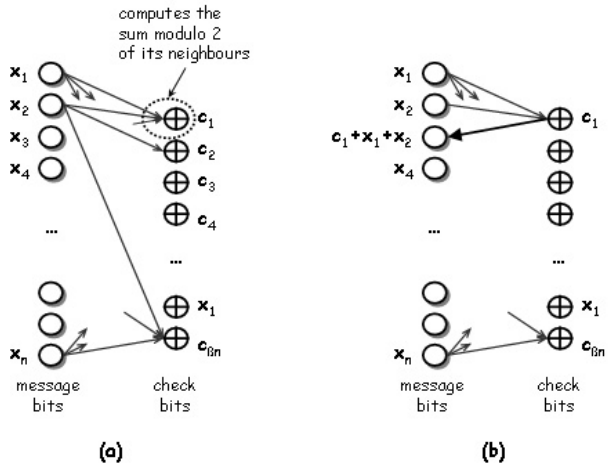


Figure V.6.: (a) A bipartite graph defines a mapping from message bits to check bits. (b) Bits $x_1$, $x_2$, and $c$ are used to solve for $x_3$.

construct the original messages. A viable solution consists in finding an alternative route as soon as possible. Authors of [KFC04] have also integrated route fixing into the framework with encoding and re-transmission of data and have used the Beacon Vector Routing (BVR) protocol [FRZ+05] (a particular case of geographic routing for WSNs) at the routing layer to provide the special support needed. Results show improvements in both reliability of transmissions and RTTs experienced by messages.

## 3.2. Tornado Codes

Tornado Codes were introduced in 1997 to be fast, loss resilient codes, suitable to error-prone transmission channels. A preliminary version of these codes was presented in [LMS+97] and further improvements were later added in [LMSS01]. The overall structure of Tornado Codes is related to the low-density parity-check codes introduced by Gallagar in [Gal63]. Tornado codes cannot overtake the information-recovery capacity of RS codes, which is already optimal, but focus on

faster encoding and decoding times. Namely, given $n$ the length of the output block of the encoder corresponding to an input block of length $k$ (with $k < n$), encoding and decoding times of Tornado codes are of order $O(n \ln(\frac{1}{\varepsilon}))$ for some $\varepsilon > 0$ (instead of $O(n^2)$ as in RS-codes). The price paid for much faster encoding and decoding times, with respect to RS codes, is that arrival of $k$ packets is no longer sufficient for the receiver to reconstruct the source data.

### 3.2.1. Encoding Process

Suppose the source data to be encoded is a vector of $k$ symbols in the finite field $GF(q)$ and let a symbol be a *bit*. The encoding process to yield the correspondent Tornado Code $C$ relies on a bipartite graph $B$. Let the overall code be referred to as $C(B)$. It consists of a *systematic code* and thus includes the $k$ bits of the original source message, named *message bits*, as well as $\beta k$ redundant bits ($0 < \beta < 1$) named *check bits*. The bipartite graph consists of $k$ left nodes, $\beta k$ right nodes, and some edges to connect left nodes to right nodes. Left nodes correspond to message bits and right nodes correspond to check bits. Check bits are obtained by XOR-ing over the source message bits according to the bipartite graph $B$. Namely, a check bit is worked out by XOR-ing over all the message bits it is connected to in the graph $B$ (see Fig. V.6(a)). The bipartite graph is sparse, random and irregular. Its construction is actually a bit complex, and is principally oriented to the reduction of decoding costs (see below). By making things a bit more complex than has been described so far, a Tornado code is generally obtained with a multi-bipartite graph, i.e., a sequence (cascade) of successive bipartite graphs. In the first stage, a set of $\beta k$ redundant bits is generated from $k$ message bits with a first bipartite graph, as explained above. Then, a second graph is added starting from the check bits of the previous graph. The new graph has thus $\beta k$ left bits and $\beta^2 k$ right bits. Then, a further graph can be added with $\beta^2 k$ left bits and $\beta^3 k$ right bits, and so on. More formally, it should be said that a set of codes $C(B_0),\ C(B_1),\ \ldots,\ C(B_m)$ is constructed by means of a set of bipartite graphs $B_0,\ B_1,\ \ldots,\ B_m$ where a generic graph $B_i$ has $\beta^i k$ left nodes and $\beta^{i+1} k$ right nodes. The value $m$ is chosen such that $\beta^{m+1} k$ is roughly $\sqrt{k}$. The check bits of the last graph are finally encoded via another erasure code, say $C$, which can be, for example, a Reed-Solomon code

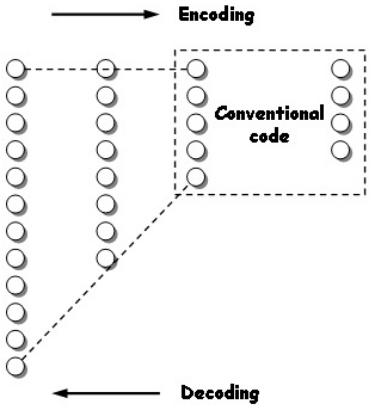


Figure V.7.: The code levels.

as proposed in [LMSS01]. In Fig. V.7 the last level erasure code is a systematic code with left nodes as original data and right nodes as redundant data. The final code includes the original $k$ message bits, the check bits produced by the bipartite graphs at the $m$ different levels, and the last-level RS-code. The entire code can be referred to as $C(B_0,\ B_1,\ \ldots,\ B_m,\ C)$. Given $k$ the number of original message bits, the total number of check bits is as follows.

$$\sum_{i=1}^{m+1} \beta^i k + \frac{\beta^{m+2} k}{1-\beta} = k\beta(1-\beta) \tag{V.9}$$

Hence, the code $C(B_0,\ B_1,\ \ldots,\ B_m,\ C)$ has globally $k$ message bits and $k\beta(1-\beta)$ check bits.

### 3.2.2. Decoding Process

Decoding relies on the assumption that the receiver knows the position of each received symbol within the graph and is performed going backward throughout the multi-bipartite graph. The RS-code $C$ is decoded first. If the decoding process tackles to reconstruct all the original $\beta^{m+1}k$ bits of the RS-code then the last level of check bits is complete whereas the last but one level as well as all the previous

ones still need reconstruction. The decoding process proceeds one level at a time and requires finding each time a check bit on the right side such that only one adjacent message bit (among those that have generated it) is missing. The missing message bit can thus be obtained by XOR-ing among the check bit and its known message bits. As an example (see Fig. V.6(b)), suppose $c$ is a check bit whereas $x_1,\ x_2$ , and $x_3$ the message bits it depends on i.e., $c = x_1 \oplus x_2 \oplus x_3$. Suppose also that, after transmission, $c$, $x_1$, and $x_2$ are known whereas $x_3$ is unknown. Then, the following holds true.

$$x_1 \oplus x_2 \oplus c = (x_1 \oplus x_2) \oplus (x_1 \oplus x_2) \oplus x_3 = x_3 \tag{V.10}$$

The missing message bits are yielded one at a time by exploiting the graph relations in which they are involved together with other known message bits, to produce other known check bits. Once decoded a single level, the previous one is decoded then. By repeating the same procedure for each bipartite graph, the original *k*-bit message can finally be reconstructed, as well.

### 3.2.3. Properties of Tornado Codes

This sub-section provides some insights into interesting features of Tornado codes related to constraints and requirements of the encoding process, and some advantages and drawbacks too.

**Property 1**

The first property is about the feasibility of Tornado codes. This is strictly connected to the particular characteristics of the multi-bipartite graph in that successful decoding is only possible if an appropriate *degree distribution* is fulfilled during construction of the graph. The degree of a node belonging to the *k*-th graph level gives the number of nodes of the *(k-1)*-th level that node is generated by. [LMSS01] makes use of linear programming techniques to select the degrees to assign to the nodes of the same level and specifically defines appropriate *sequences* of node degrees.

**Property 2**

The shape of the multi-bipartite graph also impacts the performance of Tornado

codes. Both encoding and decoding times strictly depend on the number of edges which are present in the multi-bipartite graph. Clearly, the higher is the number of edges, the higher is the computational burden of the code. The aforementioned encoding approach based on linear programming converges in *linear* time and also allows linear time decoding, as has been formally demonstrated in [LMSS01]. Limiting encoding and decoding complexity to linear time is also the reason why the last bipartite graph in the cascading sequence of bipartite graphs should have $\sqrt{k}$ right nodes ($k$ being the number of nodes associated with the original message) and the total number of graphs in the sequence should consequently be $O(\log k)$. $\sqrt{k}$ nodes are the ingress nodes of the last stage of encoding and, given the quadratic complexity of the RS-code which is used, by reducing the input size of the encoder, also the complexity is limited. The overall time complexity is about linear.

**Property 3**

*Code efficiency* of Tornado codes is worse than code efficiency of RS-codes. This is because Tornado codes need slightly more than $k$ encoded bits to arrive to destination for successful decoding of the original $k$ message bits. Namely, $1.063k$ encoded bits, at least, must be collected at destination. However, better results have been obtained with an improved version of Tornado codes. This new version uses a three-level graph and does not perform RS-coding at the last level of the graph but rather repeats the same type of random bipartite graph as the previous two levels. Results show that $1.033k$ encoded bits are needed with this version of Tornado codes to reconstruct the original $k$ message bits.

**Property 4**

Some of the limits of Tornado codes are related to the *stretch factor* which is defined as the ratio $n/k$. Tornado codes can admit only a small stretch factor so as to limit the computational burden. A typical value is for example $4$. However, a limited stretch factor limits in practice the applicability of Tornado codes. Another drawback related to the stretch factor is that, both when using RS code or another kind of erasure code at the last stage, the stretch factor must be *predetermined* before encoding takes place. Hence, researchers need to estimate, in advance, the loss probability of the channel to determine the amount of redundancy to put into the data. For most Internet applications, the loss probability varies widely

over time. Thus, in practice, the loss rate has to be overestimated, making the codes slower to decode and far less efficient in transmitting information. If the loss probability is underestimated instead, the code fails.

## 3.3. Luby Transform Codes

With Luby-Transform codes (LT codes for short) it is no more necessary to tune the encoding process over the loss characteristics of the transmission channel. LT codes do not rely on a fixed rate to add redundancy to the data to send out, rather they are said to be *rateless* codes. They were first devised in $1998$ and can definitely be considered the first full realization of the concept of digital fountains [Lub02]. Let a symbol be an $l$-bit string. Encoding is performed over $k$ input symbols and has the potential to generate a limitless number of encoded symbols. Each encoded symbol is produced, on average, by $O(\ln(k/\delta))$ symbol operations, being $0 < \delta \leq 1$. The original $k$ input symbols can be reconstructed with probability $(1-\delta)$ from any set of $k + O(\sqrt{k}\ln^2(k/\delta))$ encoded symbols out of the total number of encoded symbols produced. The total number of operations needed to recover the original $k$ symbols is about $O(k\ln(k/\delta))$. Hence, encoding and decoding times depend on the ingress data length independent of how many encoded symbols are generated and sent by the decoder. The same relations hold for the encoder and decoder memory usage. Therefore, given the reduced overall costs of LT codes with respect to the other aforementioned erasure codes, encoding can be performed over bigger chunks of data than those used in RS-codes and Tornado Codes, or even over the entire file at once. Luby-Transform codes are similar in construction and properties to Tornado codes because they also rely on a graph and the encoded symbols are obtained by XOR-ing over other initial (or previously encoded) symbols (see Fig. V.8). However, the graph construction for LT codes is quite different from Tornado codes and the resulting graph has logarithmic density. Moreover, LT codes are not systematic codes.

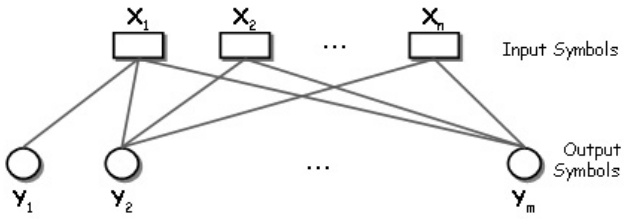


Figure V.8.: Bipartite graph illustrating input symbols and output symbols.

### 3.3.1. Encoding Process

Let the input file to be encoded have length $N$. The file is partitioned into $k = N/l$ input symbols i.e., each input symbol has length $l$. Let an input symbol be referred to as a packet. Similarly, each output symbol is referred to as an encoded packet. The encoding operation defines a bipartite graph connecting output symbols to input symbols. For each output symbol (encoded packet) encoding is performed by i) randomly choosing the degree $d$ from a given degree distribution, ii) choosing, uniformly at random again, $d$ distinct input symbols as neighbours[4] of the encoded symbol, and iii) XOR-ing over the selected $d$ input symbols taken as neighbours. The result of the bit-by-bit XOR among the selected symbols is the value of the encoded symbol. The degree distribution is computed easily independently of the data length (see below). To work properly, the decoder should be able to reconstruct the bipartite graph without errors, i.e., it needs to know which $d$ input symbols have been used to generate any given output symbol, but not their values.

### 3.3.2. Decoding Process

The decoder needs to know the degree and set of neighbours of each encoded symbol. Hence, for example, the degree and list of neighbour indices may be given explicitly to the decoder for each encoded symbol, or, as another example, a key may be associated with each encoded symbol and then both the encoder

[4] The neighbours of an encoded symbol are those symbols of the previous layer of the bipartite graph to which the encoded symbol is connected by edges. An encoded symbol is obtained by XOR-ing over all its neighbours of the previous graph-layer

and the decoder apply the same function to the key to produce the degree and set of neighbours of the encoded symbol. The encoder may randomly choose each key it uses to generate an encoded symbol and keys may be sent to the decoder along with the encoded symbols. The decoder receives $K$ output symbols and the bipartite graph, from which it tries to recover the input symbols. Typically, $K$ is slightly larger than $k$. The decoding process can be described as follows (see the Belief Propagation algorithm [RSU00]).

1. Find an output symbol $y_i$ that is connected to only one input symbol $x_j$. If there is no such output symbol, the decoding algorithm halts at this point, and fails to recover all the input symbols.

2. Set $x_j = y_i$.

3. Add (i.e., perform a bit-by-bit XOR) $x_j$ to all the output symbols that are connected to $x_j$.

4. Remove all the edges connected to the input symbol $x_j$.

5. Repeat until all $\{x_j\}$ are determined or otherwise no more output symbols can be found that have exactly one neighbour. In the first case decoding ends successfully, in the latter, it fails.

LT codes require a simple decoder, but it turns out that the *degree distribution* is a critical part of the design. The decoding process as described above will not even start if there is no output symbol of degree 1. This means that a good degree distribution is required for good decoding performances. Indeed, whether or not the decoding algorithm is successful depends solely on the degree distribution. The degree distribution used to produce LT codes is the *Soliton Distribution* or the more efficient *Robust Soliton Distribution*. Further details on the degree distribution analysis and the Soliton and Robust soliton distributions can be found in [Lub02].

### 3.3.3. Applications

In 1998 a new company (Digital Fountain Inc) was launched by the inventor of LT codes (Michael Luby) [Rob02]. The company aimed at exploiting the digital fountain's technology as a tool for downloading popular software packages, such as Adobe Acrobat, and indeed, Adobe was an early investor, as were Cisco Systems and Sony Corporation. Digital Fountain Inc. took obviously a patent on LT codes and this has been limiting utilization of LT codes in the research community as well as in other research and development centres. Digital Fountain's researchers have been working on a new generation of codes and the group is also addressing other related research issues, such as application of the technology to video-on-demand and streaming media. Recently, the group has been focusing on congestion controls to ensure that the Digital Fountain packets behave in ways that don't cause network flow problems. Indeed, security issues are being addressed in collaboration with some universities to conceive, for example, a packet authentication system for verifying that data has come from a particular source. Launched to address the problems of streaming media and multicast, Digital Fountain -like many new technology companies- has found an unexpected niche. Without the requirement for TCP acknowledgments, data encoded with Luby transform codes travels faster than ordinary packets; this advantage becomes significant over long distances. Thus, the technology has proved particularly useful to companies that frequently transmit extremely large data files over long distances, such as movie studios and oil exploration companies. Inspired to LT codes, the work in [DPR05] proposes using digital fountains in wireless sensor networks. The sensor network is viewed as a big DataBase with $k$ independent data-generating nodes and $n$ data-storage nodes. Each storage node is assumed to have a limited storage capacity of only one single data packet. Data packets are small and generated independently from each other. It is indeed assumed that no correlation exists between them. The $k$ source data packets are encoded and distributed after addition of redundancy to the $n$ storage nodes. A collector/sink node wishing to retrieve information from the sensor network needs only to query any set of storage nodes of size slightly larger than $k$ to reconstruct the entire data sensed at the source nodes. The solution proposed is inspired to the family of protocols named Sensor Proto-

cols for Information via Negotiation (SPIN). These protocols assume that all the nodes have enough storage space to store all the information gathered in the network. Hence the information gathered by the sensors is disseminated throughout the entire network and a user can query any node to get the required information immediately by only communicating with one node. The solution in [DPR05] is different in that each sensor node has not enough memory to store all the data generated in the network and thus collects a combination of the total information gathered in the network. One single interrogation is not therefore sufficient but slightly more than $k$ are required.

## 3.4. Raptor Codes

Raptor codes are a recent extension of LT codes (2004) being conceived with the aim to improve the decoding probability of LT codes [Sho04]. The decoding graph of LT codes, in fact, needs to be of the order of $k \log(k)$ in the number of edges ($k$ being the total number of input symbols) to make sure that all the input symbols can be recovered with high probability. Raptor codes focus on relaxing this condition such that the decoding process requires a smaller number of edges to be available. Raptor codes introduce a first level of encoding where the input symbols are pre-coded and a new set of symbols comprising some redundancy is generated. The new set of symbols is then given as input to a second level of encoding that performs LT coding (see Fig. V.9). Pre-coding consists of a traditional erasure correcting code, say , with a fixed stretch factor. The choice of code depends on the specific application in mind. One possible choice is to use a Tornado code, but other choices are also possible such as, an LT-code with the appropriate number of output symbols. The second-level LT code needs appropriate tuning such that it is capable to recover all the input symbols (i.e., symbols given as input to pre-coding) even in face of a fixed fraction of erasures.

### 3.4.1. Encoding Process

Let $C$ be a linear code producing $n$ output symbols from $k$ input symbols, and let $D$ be a degree distribution. A *Raptor Code* is thus an LT code with distribution

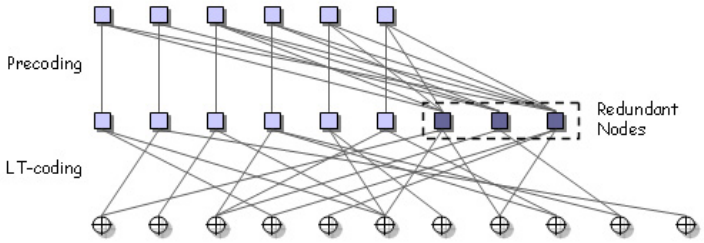


Figure V.9.: Raptor Codes: the input symbols are appended by redundant symbols (dark squares) in the case of a systematic pre-code. An appropriate LT-code is used to generate output symbols from the pre-coded input symbols.

$D$ on $n$ symbols which are the coordinates of the code-words in $C$. Code $C$ is called *pre-code* of the Raptor Code. The input symbols of a Raptor Code are thus the $k$ symbols used to construct the code-word in $C$. The code-word consists of $n$ *intermediate symbols*. Then, the output symbols are the symbols generated by the LT Code from the $n$ intermediate symbols using the degree distribution $D$. Typically, $C$ is a *systematic* code, though this is not strictly necessary. The encoding cost of a Raptor Code takes into account the encoding cost of both pre-coding and LT coding. It thus includes the number of arithmetic operations sufficient for generating a code-word in $C$ from the $k$ input symbols. The encoding cost is generally a per-symbol cost. Hence, it is worked out dividing the total cost to generate $n$ output symbols by $k$.

### 3.4.2. Decoding Process

A reliable decoding algorithm of length $m$ for a Raptor code is an algorithm which can recover the initial $k$ input symbols from any set of $m$ output symbols and errs with probability at most $1/(k^c)$ for some constant $c > 1$. The decoding process will include in the first stage an LT decoding process that will recover a fraction of the $n$ intermediate symbols that have been given as input to the LT encoder in the encoding phase. Then, the second stage will consist of a classical erasure

decoding process that should be able to recover the $k$ original input symbols of the entire Raptor encoder. The decoding cost of a Raptor Code is the expected number of arithmetic operations sufficient to recover the $k$ input symbols, divided by $k$. However, the overall efficiency of Raptor codes should not be measured only in terms of the number of operations per single input symbol but also in terms of *space* and *overhead*. Space refers to the memory consumption for storage of intermediate symbols. Overhead instead is a function of the decoding algorithm used, and is defined as the number of output symbols that the decoder needs to collect in order to recover the input symbols with high probability. Different types of Raptor Codes can be obtained depending on the type of erasure code $C$ which is used for pre-coding and the particular degree distribution used in the second level encoding. Hence, a complex pre-coding[5] can be combined with a simple degree distribution of the LT encoder or vice-versa a simple pre-coding algorithm can be combined with a complex degree distribution. In between these two extremes other solutions are also possible. Optimized Raptor Codes applied to $k$ original input symbols have the potential to generate an infinite stream of output symbols. Any subset of symbols of size $k(1+\varepsilon)$, for some $\varepsilon > 0$, is then sufficient to recover the original $k$ input symbols with high probability. The number of operations needed to produce each output symbol is of order $O(\log(1/\varepsilon))$ whereas order $O(k\log(1/\varepsilon))$ operations are needed to recover all the $k$ original data symbols. Differently from LT codes, Raptor codes can be *systematic codes*.

### 3.4.3. Applications

Raptor Codes are being used in commercial systems by Digital Fountain Inc. for fast and reliable delivery of data over heterogeneous networks. The Raptor implementation of Digital Fountain reaches speeds of several gigabits per second, on a $2.4$ GHz Intel Xeon processor, while ensuring very stringent conditions on the error probability of the decoder.

[5] Sometimes complex erasure codes can be used when intermediate symbols (code-words in $C$) can be calculated off-line via pre-processing.

## 4. Network Coding

Network Coding is recently emerging as a new promising research field in the networking framework [ACLY00]. Although sharing some basic concepts with erasure coding and digital fountains, network coding differs from the aforementioned encoding techniques in many interesting aspects. First of all, the target of network coding is something different from that of digital fountains. Network coding has been conceived to allow *better resource utilization* during transmission over the network, especially as regards *bandwidth* and *throughput*. In addition, network coding allows *optimal delays* and better traffic distribution throughout the network thus leading to overall *load balancing* which is very promising in scenarios where power saving is a major concern (see e.g., wireless ad hoc and sensor networks). It adds *robustness* against node and connections failures (even permanent) as well as *flexibility* against changes in the network topology [KM02]. Network Coding therefore is not concerned with reliable transmissions, which is instead the major target of Erasure Coding, but focuses more generally on the overall optimization of network usage. Moreover, as far as robustness and flexibility are concerned, network coding seems to be quite suitable to new emerging scenarios of extreme and challenged networks like, for example, Delay Tolerant Networks (DTNs), intermittently connected ad hoc and sensor networks, underwater acoustic sensor networks, and vehicular ad hoc networks. These scenarios are prone to link failures because nodes can move, crash or simply go to sleep-mode to save energy. It has been shown that network coding-based algorithms for data dissemination where all nodes are interested in knowing all the information produced, are actually very efficient. Apart from targets and performance, the other great difference between erasure and network coding is recursive coding at intermediate nodes. With network coding, both source nodes and intermediate/relay nodes encode the original and/or to-be-forwarded data while in generic relay systems relay nodes simply repeat data towards the next identified hops (see Fig. V.10). To illustrate the advantages of network coding let consider a classical example. Fig. V.11 shows the graph representation of a one-source two-sink network. The capacity of each link (edge) of the network is one bit per second. The value of the max-flow from the source node to each of the destination nodes $t_1$ and

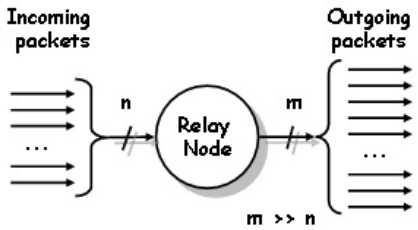


Figure V.10.: Relay nodes perform encoding over $n$ ingress data packets and produce $m$ data packets to send out. The number of encoded packets is greater than the number of incoming data packets.

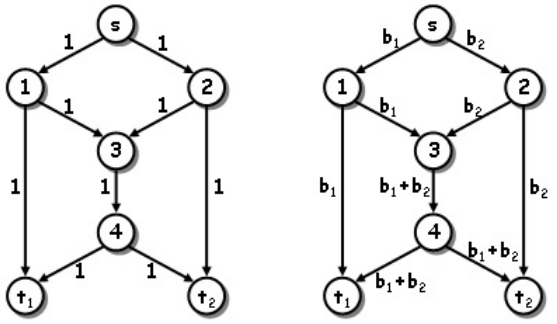


Figure V.11.: A one-source two-sink network with coding: link capacities (left) and network coding scheme (right).

$t_2$, according to the well known max-flow min-cut theorem is 2. So it should be possible to send 2 bits, $b_1$ and $b_2$, to $t_1$ and $t_2$ simultaneously. However, this is only possible by using network coding. Fig. V.11 shows the network coding scheme that allows the max-flow. "+" denotes XOR operation. Instead of forwarding $b_1$ and $b_2$ in two separate transmissions, node 3 encodes them and sends just a single encoded packet, $b_1 + b_2$, which eventually reaches $t_1$ and $t_2$ via node 4. As node $t_1$ and $t_2$ also get respectively $b_1$ and $b_2$ via different routes, they are able to recover the missing bit from the encoded packet received from node 3. The above network scheme is well known and called "butterfly network". It was presented in 2000 in [ACLY00], when the term network coding was originally proposed. The same article also demonstrated that network coding is necessary to achieve the theoretical upper bound limit for throughput utilization (max-flow) in optimal multicast schemes that include two or more destination nodes. However, throughput benefits of network coding are not only limited to multicast flows [NMY03] and can extend to other traffic patterns, e.g., unicast transmissions [NMY03]. Throughput gains are different depending on the network graph. Specifically, they can be very significant in directed graphs whereas, in undirected graphs (e.g., a wired network where all links are half-duplex) the throughput gain is at most a factor of two [CFS06] [LL04]. In the following sections 4.1 and 4.2, some insights on the encoding and decoding processes of network coding will be given. Indeed, a particular type of network coding will be described which is called *random*, *linear network coding* [FLBW06]. Linear coding is one of the simplest coding schemes for network coding. It regards a block of data as a vector over a certain base field and allows a node to apply a linear transformation to this vector before forwarding it [LYC03]. When the transformation is performed at each node independently from other nodes, and using random coefficients, linear coding is said to be random network coding. Its efficacy has been extensively studied, mainly analytically, for different types of networks corresponding to different graphs [SET03], [HKM+03], and [HMS+03]. Cases that have been studied include: i) one-source networks and networks with multiple sources either independent or linearly correlated; ii) acyclic networks and cyclic networks; iii) delay-free and delayed networks. In all these environments random network coding has brought interesting and valuable benefits. Finally, some recent applications of Network Coding will be described in

Section 4.3.

## 4.1. Encoding Process

As mentioned before, encoding is performed both at the source and intermediate relay nodes which are visited by the data flowing over the network. Data is organized into packets. Assume that all the incoming/generated *packets* have the same length of *L bits*. If a packet is shorter, then it is padded with trailing 0s. The content of a packet can be interpreted as a sequence of *symbols*, where each symbol corresponds to a sequence of $s$ consecutive bits. Hence, each *0*-padded packet, *L*-bit long, is a sequence of $L/s$, say $l$, symbols. Each symbol belongs to the field $GF(2^s)$ since it can be represented over $s$ bits. In Linear Network Coding the encoding process works out a linear combination over a set of incoming packets (at a relay node) or original packets (at the source) and produces new packets for sending out, which have the same size ($L$ bits) of the packets that have been combined together. The arithmetic operations involved in linear combinations (addition and multiplication) are performed over the field $GF(2^s)$. Assume having a set of $n$ ingress packets, namely $M^0,\ M^1,\ ,\ \ldots,\ M^{n-1}$. Each packet is a vector of $l = L/s$ symbols. Ingress packets are *original packets* in case of source nodes, whereas, in case of relay nodes, they may be original and *incoming packets* as well. Incoming packets are intended to come from one or even multiple nodes. However, suppose for simplicity that the incoming packets come all from the same source node. The node that performs encoding starts generating $n$ coefficients, one for each ingress packet: $g_0,\ g_1,\ ,\ \ldots,\ g_{n-1}$. Coefficients are symbols, i.e., $g_i \in GF(2^s)$. Then the node generates an outgoing packet $X$ of size $L$ bits as follows.

$$X = \sum_{i=0}^{n-1} g_i M^i \tag{V.11}$$

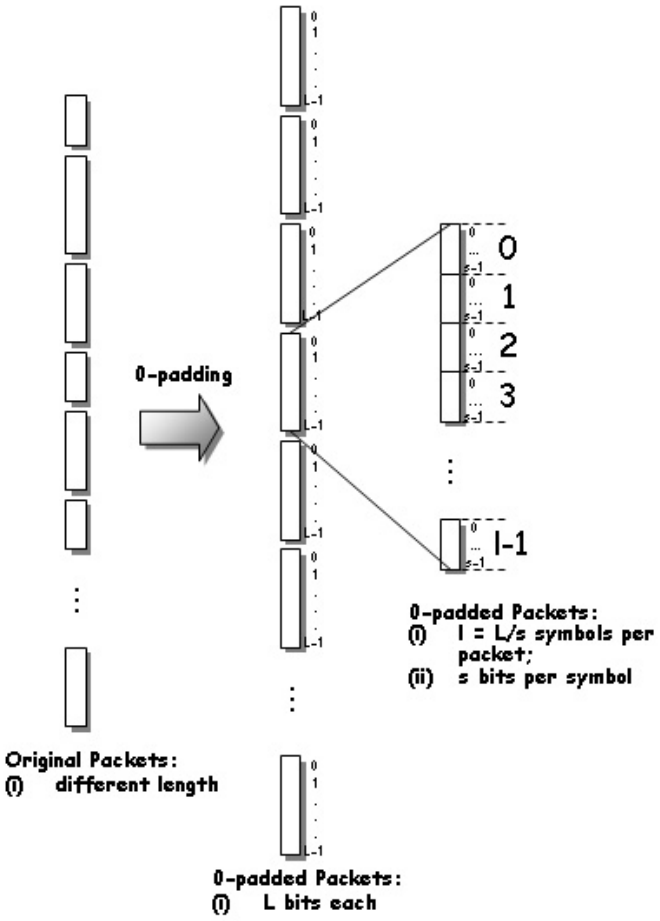


Figure V.12.: All the packets have the same length of $L$ bits. Shorter packets are *0*-padded.

With much detail the encoding process for a single outgoing packet $X$ works as follows.

$$X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ \dots \\ x_{l-1} \end{pmatrix} = \sum_{i=0}^{n-1} g_i M^i = \sum_{i=0}^{n-1} g_i \cdot \begin{pmatrix} m_0^i \\ m_1^i \\ m_2^i \\ \dots \\ \dots \\ m_{l-1}^i \end{pmatrix} =$$

$$g_0 \cdot \begin{pmatrix} m_0^0 \\ m_1^0 \\ m_2^0 \\ \dots \\ \dots \\ m_{l-1}^0 \end{pmatrix} + g_1 \cdot \begin{pmatrix} m_0^1 \\ m_1^1 \\ m_2^1 \\ \dots \\ \dots \\ m_{l-1}^1 \end{pmatrix} + g_2 \cdot \begin{pmatrix} m_0^2 \\ m_1^2 \\ m_2^2 \\ \dots \\ \dots \\ m_{l-1}^2 \end{pmatrix} + \ldots + g_{n-1} \cdot \begin{pmatrix} m_0^{n-1} \\ m_1^{n-1} \\ m_2^{n-1} \\ \dots \\ \dots \\ m_{l-1}^{m-1} \end{pmatrix} \tag{V.12}$$

where $x_i,\ m_i^j,\ g_j \in GF(2^s)$. $x_i$ is the *i*-th symbol of the egress vector (packet) $X$. $m_i^j$ is the *i*-th symbol of the *j*-th ingress packet $M^j$. $g_j$ is the coefficient that multiplies the *j*-th ingress packet $M^j$. The same encoding process is then repeated more times. Specifically, from $n$ ingress packets a node generates $m$ outgoing, egress packets with $m \gg n$ so as at the receiving site decoding is possible after having received *any* $n$ *out of* $m$ packets. Hence, the encoding node actually generates $m$ sets of coefficients, one for each egress packet to generate, and then performs encoding as follows (all the three formulas below are equivalent).

$$X^j = \sum_{i=0}^{n-1} g_i^j M^i \quad j = 0,\ \ldots,\ m-1 \tag{V.13}$$

$$\begin{pmatrix} X^0 \\ X^1 \\ X^2 \\ \cdots \\ \vdots \\ \cdots \\ X^{m-1} \end{pmatrix} = \begin{pmatrix} g_0^0 & g_1^0 & g_2^0 & \cdots & g_{n-1}^0 \\ g_0^1 & g_1^1 & g_2^1 & \cdots & g_{n-1}^1 \\ g_0^2 & g_1^2 & g_2^2 & \cdots & g_{n-1}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_0^{m-1} & g_1^{m-1} & g_2^{m-1} & \cdots & g_{n-1}^{m-1} \end{pmatrix} \begin{pmatrix} M^0 \\ M^1 \\ M^2 \\ \vdots \\ M^{n-1} \end{pmatrix} \tag{V.14}$$

$$X_{(m\times 1)} = G_{(m\times n)} \cdot M_{(n\times 1)} \tag{V.15}$$

where $X^i$ is the *i*-th egress packet that is generated and $g_i^j$ is the *i*-th symbol of the *j*-th encoding vector, i.e., the *j*-th set of coefficients which is generated to produce the *j*-th encoded packet. All the cofficients $g_i^j$ together form the $(m \times n)$-sized encoding matrix $G$. Each encoded data packet $X^j$ is named *information vector* and the corresponding vector of coefficients it is obtained from, $g^j$, is said to be an *encoding vector*. The encoding node produces and sends out packets by including both encoding and information vectors, thus an outgoing packet is actually a tuple $< g^j,\ X^j >$ for $j = 0,\ 1,\ \ldots,\ m-1$. Each encoding vector includes $n$ symbols, whereas each information vector includes $l$ symbols, as many as in the ingress data items $M^i$ $(i = 0,\ 1,\ \ldots,\ n-1)$.

$$< g^j,\ X^j > = < (g_0^j,\ g_1^j,\ g_2^j,\ \ldots,\ g_{n-1}^j),\ (x_0^j,\ x_1^j,\ x_2^j,\ \ldots,\ x_{l-1}^j) > \tag{V.16}$$

When an encoding vector includes all zeros but one single one in the *i*-th position, it means that the *i*-th ingress data item is not encoded.

$$g^i = e^i = (0,\ 0,\ \ldots,\ 0,\ 1,\ 0,\ \ldots,\ 0) \quad \Rightarrow \quad X^i = M^i \tag{V.17}$$

### 4.1.1. Recursive Encoding

Suppose that the following conditions hold.

(i) An intermediate relay receives and stores a set of $r$ tuples, as follows.

$$< g^0,\ X^0 >,\ < g^1,\ X^1 >,\ < g^2,\ X^2 >,\ \ldots,\ < g^{r-1},\ X^{r-1} > \qquad \text{(V.18)}$$

(ii) All the information vectors received $(X^0,\ X^1,\ \ldots,\ X^{r-1})$ come from the same set of original data $(M^0,\ M^1,\ \ldots,\ M^{n-1})$ produced by a single source node somewhere.

Then, the intermediate node performs re-encoding on the set of information vectors that it has received: $X^0,\ X^1,\ \ldots,\ X^{r-1}$. It is not necessary that $r \gg n$ because the intermediate node does not need decoding at this stage. It simply re-encodes the pieces of data it has received. This results in adding some more redundancy into the network for some part of the original data. Re-encoding is performed by generating a certain number of encoding vectors, as many as the number of information vectors that the node wants to send out. Suppose, for example, that the node wants to generate $k$ information vectors. It should be noted that there is not a mathematical constraint on the values allowed for $r$ and $k$. Rather, the right tuning of these parameters depends on the networking scenario and is still a challenging open issue. After having chosen the appropriate value for $k$, the relay node acts as follows.

(a) Selects $k$ new encoding vectors $h^j$-s $(j = 0,\ 1,\ \ldots,\ k-1)$. Each encoding vector includes as many symbols as the number of ingress information vectors the new encoding is applied to. In this case each encoding vector is *r*-symbol long.

$$h^j = (h_0^j,\ h_1^j,\ h_2^j,\ \ldots,\ h_{r-1}^j), \quad j = 0,\ 1,\ \ldots,\ k-1 \qquad \text{(V.19)}$$

(b) Produces $k$ new information vectors as follows.

$$Y^j = \sum_{i=0}^{r-1} h_i^j X^i, \quad j = 0,\ 1,\ \ldots,\ k-1 \qquad \text{(V.20)}$$

The new encoding matrix is therefore as follows.

$$H_{(k\times r)} = \begin{pmatrix} h_0^0 & h_1^0 & \dots & h_{r-1}^0 \\ h_0^1 & h_1^1 & \dots & h_{r-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{k-2} & h_1^{k-2} & \dots & h_{r-1}^{k-2} \\ h_0^{k-1} & h_1^{k-1} & \dots & h_{r-1}^{k-1} \end{pmatrix} \tag{V.21}$$

Whereas the overall re-encoding process can be represented as follows.

$$\begin{pmatrix} Y^0 \\ Y^1 \\ \vdots \\ Y^{k-2} \\ Y^{k-1} \end{pmatrix} = \begin{pmatrix} h_0^0 & h_1^0 & \dots & h_{r-1}^0 \\ h_0^1 & h_1^1 & \dots & h_{r-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{k-2} & h_1^{k-2} & \dots & h_{r-1}^{k-2} \\ h_0^{k-1} & h_1^{k-1} & \dots & h_{r-1}^{k-1} \end{pmatrix} \begin{pmatrix} X^0 \\ X^1 \\ \vdots \\ X^{r-1} \end{pmatrix} =$$

$$= \begin{pmatrix} h_0^0 & h_1^0 & \dots & h_{r-1}^0 \\ h_0^1 & h_1^1 & \dots & h_{r-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{k-2} & h_1^{k-2} & \dots & h_{r-1}^{k-2} \\ h_0^{k-1} & h_1^{k-1} & \dots & h_{r-1}^{k-1} \end{pmatrix} \cdot \begin{pmatrix} g_0^0 & g_1^0 & g_2^0 & \dots & g_{n-1}^0 \\ g_0^1 & g_1^1 & g_2^1 & \dots & g_{n-1}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_0^{r-1} & g_1^{r-1} & g_2^{r-1} & \dots & g_{n-1}^{r-1} \end{pmatrix} \cdot$$

$$\cdot \begin{pmatrix} M^0 \\ M^1 \\ M^2 \\ \vdots \\ M^{n-1} \end{pmatrix} \triangleq G'_{(k\times n)} \cdot \begin{pmatrix} M^0 \\ M^1 \\ M^2 \\ \vdots \\ M^{n-1} \end{pmatrix} \tag{V.22}$$

where $G'_{(k\times n)}$ is equal to $H_{(k\times r)} \cdot G_{(r\times n)}$ and is hereafter called *re-encoding* matrix.

(c) Produces the $k$ final encoding vectors to include in the outgoing tuples. They

correspond to the rows of $G'$ worked out as follows.

$$g'^{j}_{i} = \sum_{t=0}^{r-1} h^{j}_{t} \cdot g^{t}_{i}, \quad i: 0, \ldots, n-1, \ j: 0, \ldots, k-1, \ t: 0, \ldots, r-1 \quad \text{(V.23)}$$

(d) Sends out $k$ tuples as follows.

$$< g'^{0}, Y^{0} >, < g'^{1}, Y^{1} >, < g'^{2}, Y^{2} >, \ldots, < g'^{k-1}, Y^{r-1} > \quad \text{(V.24)}$$

This process is equivalent to *encoding only once* the original data $(M^0, M^1, \ldots, M^{n-1})$ with the new set of encoding coefficients. At the receiver site the number of incoming tuples to collect prior to decoding corresponds to the number of symbols included in a single encoding vector or, from another standpoint, to the number of original data packets (as can be seen, it is always $n$). Finally, it should be noticed that the outgoing information vector is again $l$-symbol long, as follows.

$$Y^{i} = (y^{i}_{0}, y^{i}_{1}, y^{i}_{2}, \ldots, y^{i}_{l-1})^{T} \quad \text{(V.25)}$$

The computational cost of performing re-encoding at each intermediate node depends on the following operations.

(i) The product between the new encoding matrix, $H_{(k \times r)}$, and the old encoding matrix, $G_{(r \times n)}$, to obtain the final encoding matrix to be included in the outgoing tuples: $G'_{(k \times n)} = H_{(k \times r)} \cdot G_{(r \times n)}$.

(ii) The product between the new encoding matrix, $H_{(k \times r)}$, and the matrix of encoded data, $X_{(r \times l)}$, to obtain the outgoing information vectors $Y_{(k \times l)}$.

#### 4.1.2. How to choose encoding vectors?

The choice of coefficients to include in encoding vectors is, as can easily be expected, quite critical. An inaccurate choice can produce tuples which are linearly dependent on each other. When the destination node receives a tuple which is linearly dependent on another tuple already received, it discards the new tuple

because it does not contain innovative information and is thus of no use (see Section 4.2 on the Decoding Process for further details). This directly impacts the decoding time causing the destination node to wait for more packets than strictly necessary to be able to do the decoding. A simple algorithm to generate such coefficients is proposed in [HKM+03] and gives rise to the so-called *Random Linear Coding*. Each node in the network selects *uniformly at random* its coefficients over the field $GF(2^s)$ in a completely *independent* and *decentralized* manner. Simulation results indicate that even for small field sizes (e.g., with $s = 8$), the probability of selecting linearly dependent combinations is *negligible*. Other approaches have been proposed in [SET03] and [FS04]. The first one refers to a centralized algorithm where a single entity decides which coefficients each node in the network has to assign. The latter describes deterministic decentralized algorithms that apply to restricted families of network configurations.

### 4.1.3. Generations

*Generation* is the name assigned to a set of vectors (packets) which are encoded together at a source node. Typically, in fact, a source node produces a continuous flow of packets but obviously it can not apply encoding to them all at the same time because otherwise i) it should wait for the entire packet flow to be generated to perform encoding thus causing long delay to transmission, and ii) it should generate far long encoding vectors to include in outgoing packets. This would cause severe overhead in transmission. Therefore, source vectors are grouped into generations and each generation has a corresponding matrix (that includes all the encoding vectors). To allow successful decoding at destination, it is necessary that only vectors (packets) belonging to the same generation are combined. This implies that packets should carry some information about the generation they belong to or, at least, this should be deducible in some other way so as intermediate nodes do not encode together packets of different generations thus making them impossible to decode.

## 4.2. Decoding Process

In Network Coding decoding is only needed at destination. Suppose $n$ source data packets have been generated in the network. Let them be $(M^0,\ M^1,\ \ldots,\ M^{n-1})$, with $M^i = (m_0^i,\ m_1^i,\ \ldots,\ m_{l-1}^i)$. Suppose that the receiver node has just received $r$ data packets containing $r$ tuples:

$$< g^0,\ X^0 >,\ < g^1,\ X^1 >,\ \ldots,\ < g^{r-1},\ X^{r-1} > . \tag{V.26}$$

Summing up, the encoding vectors $(g^0,\ g^1,\ \ldots,\ g^{r-1})$ are such that $g^i = (g_0^i,\ g_1^i,\ \ldots,\ g_{n-1}^i)$ whereas the $r$ encoded data packets $(X^0,\ X^1,\ \ldots,\ X^{r-1})$ are such that $X^i = \sum_{j=0}^{n-1} g_j^i \cdot M^j$, and the following relations hold true.

$$x_z^i = \textstyle\sum_{j=0}^{n-1} g_j^i \cdot m_z^j, \quad \Rightarrow$$

$$\Rightarrow \quad \begin{pmatrix} x_0^i \\ x_1^i \\ x_2^i \\ \vdots \\ x_{l-1}^i \end{pmatrix} = g_0^i \cdot \begin{pmatrix} m_0^0 \\ m_1^0 \\ m_2^0 \\ \vdots \\ m_{l-1}^0 \end{pmatrix} + g_1^i \cdot \begin{pmatrix} m_0^1 \\ m_1^1 \\ m_2^1 \\ \vdots \\ m_{l-1}^1 \end{pmatrix} + \ \ldots\ + g_0^{n-1} \cdot \begin{pmatrix} m_0^{n-1} \\ m_1^{n-1} \\ m_2^{n-1} \\ \vdots \\ m_{l-1}^{n-1} \end{pmatrix} \tag{V.27}$$

The above relations can also be written as follows.

$$X_{(r\times 1)} = G_{(r\times n)} \cdot M_{(n\times 1)} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \begin{pmatrix} X^0 \\ X^1 \\ X^2 \\ \cdots \\ \vdots \\ \cdots \\ X^{r-1} \end{pmatrix} = \begin{pmatrix} g_0^0 & g_1^0 & g_2^0 & \cdots & g_{n-1}^0 \\ g_0^1 & g_1^1 & g_2^1 & \cdots & g_{n-1}^1 \\ g_0^2 & g_1^2 & g_2^2 & \cdots & g_{n-1}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_0^{r-1} & g_1^{r-1} & g_2^{r-1} & \cdots & g_{n-1}^{r-1} \end{pmatrix} \begin{pmatrix} M^0 \\ M^1 \\ M^2 \\ \vdots \\ M^{n-1} \end{pmatrix} \tag{V.28}$$

To go back to the original messages $M^0,\ M^1,\ \ldots,\ M^{n-1}$ it is necessary that the following conditions hold true.

(i) $r \geq n$;

(ii) The encoding matrix $G_{(r\times n)}$ has rank $n$. This happens if and only if the encoding matrix includes $n$ rows linearly independent.

Provided conditions (i) and (ii) are met, the original data $M^0,\ M^1,\ \ldots,\ M^{n-1}$ can be derived via standard linear system solution techniques. A computational-efficient way of doing this on-line can be sketched as follows. If $r$ packets have been received, a decoding matrix can be constructed as follows.

$$\left(\begin{array}{cccccc|cccc} g_0^0 & g_1^0 & g_2^0 & g_3^0 & \ldots & g_{n-1}^0 & x_0^0 & x_1^0 & \ldots & x_{l-1}^0 \\ g_0^1 & g_1^1 & g_2^1 & g_3^1 & \ldots & g_{n-1}^1 & x_0^1 & x_1^1 & \ldots & x_{l-1}^1 \\ g_0^2 & g_1^2 & g_2^2 & g_3^2 & \ldots & g_{n-1}^2 & x_0^2 & x_1^2 & \ldots & x_{l-1}^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ g_0^{r-2} & g_1^{r-2} & g_2^{r-2} & g_3^{r-2} & \ldots & g_{n-1}^{r-2} & x_0^{r-2} & x_1^{r-2} & \ldots & x_{l-1}^{r-2} \\ g_0^{r-1} & g_1^{r-1} & g_2^{r-1} & g_3^{r-1} & \ldots & g_{n-1}^{r-1} & x_0^{r-1} & x_1^{r-1} & \ldots & x_{l-1}^{r-1} \end{array}\right) \tag{V.29}$$

By gaussian elimination it is possible to reduce the system to the following one in which the original messages can be read.

$$\left(\begin{array}{c|cccc} & m_0^0 & m_1^0 & m_2^0 & \ldots & m_{l-1}^0 \\ & m_0^1 & m_1^1 & m_2^1 & \ldots & m_{l-1}^1 \\ I_{(n\times n)} & m_0^2 & m_1^2 & m_2^2 & \ldots & m_{l-1}^2 \\ & \vdots & \vdots & \vdots & \ddots & \vdots \\ & m_0^{n-1} & m_1^{n-1} & m_2^{n-1} & \ldots & m_{l-1}^{n-1} \\ \hline G'_{((r-n)\times n)} & \multicolumn{4}{c}{X'_{((r-n)\times l)}} \end{array}\right) \tag{V.30}$$

To construct this matrix dynamically, as packets arrive to the receiver, they are added to the decoding matrix row by row[6]. The *i*-th information vector $X^i$ is said

[6] A row is $< (g_0^i,\ g_1^i,\ g_2^i,\ \ldots,\ g_{n-1}^i),\ (x_0^i,\ x_1^i,\ x_2^i,\ \ldots,\ x_{l-1}^i) >$.

to be *innovative* if it increases the rank of the matrix. If a packet is not innovative, it is reduced to a row of all 0s by Gaussian elimination and is ignored. While performing gaussian elimination, as soon as the matrix contains a row of the form $< e_i,\ X >$, the node knows that the original packet $M^i$ is equal to $X$ (remember that we have defined $e_i$ in Section 4.1 as a vector that includes all zeros but one single one in the *i*-th position).

## 4.3. Applications

In [WLB05] Network Coding has been proposed to allow *efficient communication in extreme networks* i.e., in delay-tolerant networks or, more generally, intermittently connected networks. According to the forwarding scheme described, nodes do not simply forward packets they overhear but send out information that is coded over the content of several packets they have received. Simulation results show that this algorithm achieves the reliability and robustness of flooding at a small fraction of the overhead. Two interesting topics have been discussed, namely generations and redundancy. The solution for generation recognition which has been proposed in [WLB05] is as follows. Let $X^j_{(i)}$ be the *j*-th information vector that originates at the *i*-th source node $S_i$. Then a function $f(X^j_{(i)})$ determines which generation the packet belongs to and $\Gamma_\gamma = \{\ X^j_{(i)} \mid f(X^j_{(i)}) = \gamma\}$ is the set of all the source vectors of a generation $\gamma$. Namely, the generation membership is determined through hashing over the *sender address* and the *packet identifier*. A new hash function is generated whenever the matrix becomes too big. The hash function is then used at a node to determine which generation to insert a given packet into, provided the size of this generation does not exceed a certain max threshold. It has been demonstrated that managing generations through hashing works better than simply incrementing the generation index from time to time, especially in ad hoc networks. Another interesting topic raised in [WLB05] is on the forwarding strategy and specifically, on how to decide when to send a new packet. The solution that has been proposed relies on a so-called *forwarding factor* $d > 0$ and establishes that whenever an innovative packet is received or generated at a node for a given generation, this has to forward a certain number of packets depending on the value of d and on whether the node is a source or

an intermediate node. Specifically, in case of an intermediate node, it first generates $\lfloor d \rfloor$ vectors from the corresponding matrix and re-broadcast them to the neighbours, then it generates and sends a further vector with probability $d - \lfloor d \rfloor$. In case of a source node, whenever it generates a new original packet it encodes and broadcasts to the neighbours $\max(1, \lfloor d \rfloor)$ vectors. It then produces and sends out a further vector if $d > 1$ with probability $d - \lfloor d \rfloor$. The delivery policy at source nodes is obviously a bit different since at least one packet must be generated from each newly produced packet. In other words, a new packet is sent out by the source at least once. With the Network Coding model described above, a node sends out on average $dG + 1$ packets where $G$ is the maximum generation size ($G = m$ in the simplest case). Since receivers can decode all original vectors when they receive a number of innovative packets equal to the generation size, a good value for $d$ strongly depends on the number of neighbours (i.e., the node density). Similar to probabilistic routing, a high forwarding factor results in a high decoding probability at the expense of a high network load. In [GR05] and [WSb] network coding is used in a peer-to-peer content distribution network named *Avalanche*. In a peer-to-peer content distribution network a server splits a large file into a number of blocks, then peer nodes try to retrieve the original file by downloading blocks from the server but also distributing downloaded blocks among them. When using network coding, the blocks sent out by the server are random linear combinations of all the original blocks. Similarly, peers send out random linear combinations of all the blocks available to them. Network Coding in *Avalanche* minimizes download times with respect to the case it is not used. Network Coding gives the system more robustness in case the server leaves early or when peers only join for short periods. Finally, some promising application fields for Network Coding are Wireless Ad Hoc, Mesh and Sensor networks where it can contribute to achieving throughput gains and more secure systems. Network coding seems to facilitate protection from eavesdroppers, since information is more spread out and thus more difficult to "overhear". It also simplifies the protection against modified packets in a network. In a network with no additional protection, an intermediate attacker may make arbitrary modifications to a packet to achieve a certain reaction at the attacked destination. However, in the case of network coding, an attacker cannot control the outcome of the decoding process

at the destination, without knowing all other coded packets the destination will receive. Given that packets are routed along many different paths, this makes controlled man-in-the-middle-attacks more difficult.

## 5. Conclusions and Open Issues

Encoding techniques are becoming popular in the evolving Internet scenario. The advent of wireless technologies, the diffusion of an ever greater and heterogeneous number of portable devices, the diffusion of multimedia and content-distribution applications as well as their adoption over wireless networks are posing and renewing a number of networking challenges, such as reliability, scalability and efficient data distribution over wireless networks. Encoding techniques have successfully been used to increase reliability and scalability, for example in multicast scenarios, in parallel download from many sources, and over overlay networks. Moreover, they are commonly applied to multimedia streaming applications to manage reliability issues without requiring feedback channels, and minimizing synchronizations and interactions among multiple receivers, and between senders and receivers. Finally, it should be noted that the advantages of encoding are not only limited to transmission reliability and scalability but also regard security. It is much harder, for example, for the attackers to catch useful information from a network because they should have to collect a sufficiently high number of data packets that need to be decoded together and they should also know how exactly to do the decoding. The main drawback of the encoding techniques analysed in this chapter is the *computational burden* involved in both the encoding and decoding processes. Also the *delay* might be negatively impacted being increased by the time spent in encoding and decoding and the time spent to transmit the extra-information produced during encoding (encoded data has higher size than the original). However, many optimizations have been studied and implementations of encoding techniques have also been provided even for Wireless Sensor Networks. Since WSNs are particularly scarce in resources, this means that resource usage has successfully been minimized. Network Coding is a sort of generalization of encoding techniques. Most of the work conducted on network coding is

analytical, so far, since it has been introduced as a way to improve use of network bandwidth. It has been shown that network coding allows achievement of the *max* possible *flow* on the networks under investigation including many different topology patterns. Network coding has also the advantage to increase *robustness* of the network against node failures, even permanent, and *adaptability* to variations of the network topology. By disseminating information throughout the network, network coding also provides *load balancing* of the network traffic, and, in addition, it adds *reliability* to transmissions as the other encoding techniques do. The main drawback of network coding is the *computational burden* which is added not only to the source and destination nodes but also to the intermediate nodes that act as relays and provide re-encoding together with usual forwarding. Another challenge in network coding is the traffic overhead injected into the network. However specific analyses do not exist yet, neither simulative nor experimental. It can easily be envisioned that the next efforts on network coding will be both on simulation and experimentation to better understand real strengths and limits of this new technique. Other interesting challenges to be addressed in the framework of network coding are how to minimize the traffic injected into the network while still retaining sufficient redundancy in the encoding process, and how to efficiently deal with dynamically generated content.

# VI. Reliable Data Collection in Sparse Sensor Networks

## 1. Introduction

This chapter inherits and follows on from the basic concepts of both the above chapters IV and V, on *Opportunistic Networking* and *Encoding Techniques* respectively. It shows how the adoption of encoding techniques is beneficial to support communications in opportunistic network scenarios. Furthermore, since sensor nodes are key elements in most smart environments and pervasive systems, the particular case of communications in *Sparse Sensor Networks* is analysed. In these kinds of networks, sensor nodes are not densely deployed over the area of interest but are rather placed *strategically* depending on the particular purpose of the running application. This is cheaper with respect to solutions with hundreds of thousands of nodes deployed randomly in the environment and can suit a vast category of applications, especially indoor or in urban environments. Sparse deployment of nodes implies that the distance between pairs of sensor nodes does not always allow direct communication in between them. Hence, data collection from sensor nodes can only be applied in case either the nodes move and approach each other to exchange data or another node acting as sink passes by and gathers information from them. The first case, however, is quite improbable since sensor nodes are generally static and only have sensing capabilities[1]. Data collection by *mobile* sink node(s) instead, is a much more interesting and valuable solution.

The reference architecture for this chapter is the *three-tier architecture* discussed

[1] Indeed, the motion of nodes would cause energy wastage and lead to drastic decrease of the network lifetime.

in [SRJB03]. It comprises a *sensor-node plane* at the lower layer, an intermediate layer of mobile relay agents referred to as *Mobile Ubiquitous LAN Extensions (MULEs)*, and an upper layer of base stations, named *Infostations*. Sensor nodes are only responsible for data sensing and temporary storage, and are (generally) relieved from other functions concerned, for example, with data forwarding and routing. This because they may not be able to communicate with each other since out of their respective communication ranges, but also to save the most energy amount possible, since their constraints on resource usage is the highest of all tiers. Data MULEs (also known as ferries, or carriers) are capable of short-range wireless communication. They move around in the network area and collect data from the sensors they pass by. They are able to communicate with both the bottom-layer sensors for downloading the data monitored and with the upper-layer Infostations to upload the data previously collected. Data MULEs have not strict energy constraints like sensor nodes, nor they have illimited energy like Infostations which are permanently energy supplied. In fact, data MULEs have limited energy budget but *renewable* since, while moving from place to place, they can easily reach locations where they can be recharged. Infostations serve for final storage and elaboration of the data gathered at the sensor nodes.

The most interesting application scenario for this architecture is the urban environment where sensor nodes can be deployed for data monitoring purposes and data MULEs can be on city buses, taxis, cars, or they can even be carried by pedestrians, bicycles, etc. Data MULEs aren't thus necessarily aware of performing data collection since this takes place automatically when data MULEs get in proximity of sensor nodes. Finally, since data gathering from a sensor node does not rely on a single MULE, but different MULEs are expected to visit the same sensor node instead, sensors can exploit long sleep time periods to save energy. Even if a MULE passes by when the sensor is sleeping, it is highly probable that another MULE will pass afterwards.

As will be shown in this chapter, data collection in a similar environment is quite critical and many factors exist which affect reliability of data transmission and result in severe packet loss. In particular, two different kinds of reliability can be discriminated, namely *one-hop* and *end-to-end reliability*. Adoption of encoding techniques is quite promising to correct both them. In particular, erasure codes

(specifically Reed-Solomon codes) can successfully be applied to manage one-hop reliability. Network coding instead can successfully support end-to-end reliability.

## 2. Unreliability issues in Sparse Wireless Sensor Networks

The three-tier wireless-sensor-network architecture we refer to in this chapter comprises of an extremely high number of heterogeneous devices. Starting from sensor nodes going up to data MULEs and Infostations, all the nodes included in this architecture have different sizes and motion capabilities. Sensor nodes have basically monitoring duties. They are typically small in size and resource-constrained (minimal processing, storage, and communication capabilities). However, special sensor nodes with medium to large size and higher resource availability can also be present in the network depending on the particular service to be provided. Sensors are located in fixed positions in the area under study/monitoring and collect *context information* from the surrounding environment. Context information is then gathered from the upper-layer data MULEs and used to run specific applications. Generally, no communications from sensors to other sensors take place.

Data MULEs are user portable devices such as PDAs, laptops or smartphones. They are more complex and powerful devices with respect to sensor nodes and are carried by people in their everyday life. Therefore, data MULEs follow the motion of their carriers (mobile users). The mobility pattern of data MULEs can be extremely heterogeneous since they can move together with pedestrians, bicycles, motorbikes, cars, city buses, etc. They read pieces of information from sensors when passing by and finally upload them to Infostations when in proximity. *Opportunistic data forwarding strategies* are used to diffuse the information produced by sensors throughout the network. This because in a similar scenario there is no way to assume the existence of a complete end-to-end path among nodes. Hence, communications need to rely on *contact opportunities* that may arise during the motion of data MULEs. Whenever two data MULEs are in the communication range of each other they can exchange the information they carry. In addition, data MULEs belonging to the same connected ad hoc island can communicate

through legacy approaches, whenever implemented/applied.

The heterogeneity of devices together with the high and various mobility of data collecting nodes lead to an *extreme* communication environment. Data exchange can only occur during *contacts* among nodes. A contact occurs when two nodes enter the communication range of each other and can comunicate directly over a single-hop link. It can also refer to the time interval in which two nodes can communicate via other intermediate nodes over a multi-hop connection (whenever a lower-layer routing protocol supports multi-hop communications). Contact durations cannot easily be foreseen. They impact on the total amount of data that can be transferred between communicating nodes and thus on the throughput achievable. But they are also very critical for the reliability standpoint. Depending on the trajectories followed by data MULEs and on their velocity, contact times can be shorter or longer, or more or less disturbed. This directly impacts on the packet loss that is experienced in both communications from sensor nodes to data MULEs and from data MULEs to data MULEs.

Two different reliability issues can be isolated in similar environments. The first one deals with *single-hop* communications whereas the second one with *end-to-end* communications. Single-hop communications occur from sensor nodes to data MULEs and from data MULEs to data MULEs during contacts. This kind of communication depends on the contact duration, on the instantaneous distance between the communicating nodes, and on the relative velocity that characterizes the communicating peers. The end-to-end reliability is mainly affected by the forwarding strategy which is responsible for finding hop-by-hop a path towards the destination node. End-to-end reliability can be measured in terms of *delivery success rate* and also in terms of the *delay* experienced by messages flowing in the network. The delivery success rate is the probability that a message generated by a source node arrives at the final destination. The delay experience is the total time to arrive to destination.

## 2.1. One-hop reliability

Communications from sensor nodes to data MULEs are influenced by the mobility of data MULEs and also by other factors which mainly depend on the transmission technology.

Authors in [ZHKS04] have tested the communication characteristics of *Mica2 mote* sensors in a *static* environment, i.e., when both the sender and destination nodes are immobile during data transmission. They have found that the received signal strength at the receiver node is severly impacted by many factors such as the *distance* between source and destination, the *geographical orientation* from source to destination, the *battery status* of the communicating sensors, and also the inevitable *manufacturing differences* between sensors. The farther is the destination node, the worst is the quality that it perceives. The *transmission range* identifies the area around the sender node where the messages that it sends can be received successfully. Nevertheless, the transmission range can only give an *average* reference to the transmission abilities of a node. It has been found that the geographical source-to-destination orientation has a great influence on the transmission quality. So, transmissions directed to the geographical South have better quality with respect to transmissions directed to the East which are affected by higher packet loss. Another factor that influences the transmission quality is the battery status, which obviously affects the transmission power. The higher is the battery level, the better is the resulting transmission quality, and the longer is the *transmission range* of the sender. In addition, physically different nodes (with the same battery level) send signals that are perceived by the same node, and at the same distance, with different intensities. Anyway, the differences in the perceived signal strength which are due to manufacturing differences of the sender nodes are less evident with respect to the differences caused by different battery levels or different source-to-destination orientations.

Authors in [ACG+04b] have studied the influence of *atmospheric conditions* on the quality of transmissions between sensor nodes (Mica2 motes, again). They have shown that the presence of *rain* or *fog* (especially rain) has a severe impact on the transmission range of a sensor node. It may drop from 55m down to 15m about. Instead, variations in the humidity ratio at different hours of the day, under sta-

ble weather conditions are negligible. The same authors have demonstrated that also variations of the *data rate* of transmissions are definitely negligible. This is probably due to the fact that the range of possible data rates for Mica2 motes is quite limited (0.258Kbps to 12.3Kbps) with respect to the data-rate range provided e.g., in 802.11b transmissions (2Mbps up to 11Mbps). The distance of the communicating nodes from ground has shown valuable impact. The optimal case is experienced when the sending node as well as the receiver node are both positioned 1m at least from the ground. When maximizing the transmission power for Mica2 motes (5dBm) the average transmission range reaches 60-65m.

Going further in the analysis of the factors impacting the transmissions among sensor nodes, authors of [ACG+06b] have studied sensor transmissions in a mobile environment. Namely, they have studied the packet loss resulting in transmissions from a fixed sensor node to a mobile sensor node moving along a straight-line trajectory such that the receiver node maintains always the same vertical distance from the sender node, and only varies the horizontal distance. Results show that the mobility of the data collecting node shrinks the contact duration between the communicating nodes and gives rise to a complex, highly irregular profile of the packet loss over time. The quality of communications worsens when changing from a low mobility scenario where the mobile sink moves at 1m/s speed to a high mobility scenario (20 to 40 km/s). Things get even worse when reducing the *duty cycle* of the communicating peers for energy efficiency.

It should be noted that, even though these results have been obtained in a wireless sensor network environment, they are also valid for communications between other kinds of nodes in an opportunistic network scenario, hence they are also valid for transmissions from data MULE to data MULE. In particular, given the different technologies that are involved in this latter case, the values of delivery success rate, packet loss, and also delay concerned are obviously different from the case of wireless sensor networks. However, given that both the communicating peers are in motion with low to high velocity, high unreliability of the transmissions is also to be expected.

## 2.2. End-to-end reliability

Reliable end-to-end transmission between nodes in an opportunistic environment concerns with *i)* finding hop-by-hop a path towards the destination node, and *ii)* transmitting data correctly at each pair-wise contact along the path to the destination. Hence, both *one-hop reliability* issues and *forwarding strategies* are involved in providing end-to-end reliability.

As has previously been discussed in chapter IV, many forwarding strategies have been conceived for opportunistic networks in the last few years. The first attempts have principally been on *epidemic-based* solutions [VB00] that simply flood messages all over the network. Epidemic dissemination guarantees *short delays* but is extremely *resource consuming* both in terms of memory occupancy and bandwidth usage. To limit resource wastage, some optimizations have been investigated. For example, by introducing *hop count limits* the number of hops that a message is allowed to traverse throughout the network cannot overtake a maximum threshold, once it is reached the message is not further forwarded. As a consequence, the bandwidth usage is contained. Memory wastage is avoided by limiting the size of the buffer allocated for data forwarding purposes and by defining smart policies for buffer replacement when extra information needs to be stored and the buffer is already full [DFL01]. An alternative approach to limit the resource wastage involved with dissemination-based forwarding strategies is to limit the number of next-hop nodes that a single message can be sent to. This solution implies the selection of only *one* or a *few* next-hop nodes for a message. The choice of a next hop is generally based on statistics on the delivery success that a node can guarantee in keeping *custody* of a message. The delivery success rate that a relay node (next hop) guarantees to transmissions towards a destination node can be worked out in many different ways based on past encounters or visits to particular places [BBL05] [LDS03], on specific context information [MHM05], etc. Obviously, the choice of a particular forwarding strategy strictly depends on the specific application which has to exploit it (e.g., if the application is more sensitive to delays, then a dissemination-based data forwarding strategy is preferable to a more selective one which selects only a single next-hop node) and also on the specific communication pattern that characterises the environment (e.g., in a more dy-

namic environment with frequent encounters among nodes even a selective data forwarding strategy is suitable). However, given the extremely variable conditions that usually characterise opportunistic environments, data forwarding strategies that provide some degree of redundancy, even minimum, and hence that select more than one single next-hop node for transmission, are preferable and result in a more robust communication pattern.

Most of the work conducted on routing and forwarding strategies in opportunistic networks relies on simulation analyses. This unfortunately has been shown not to be sufficient [CHC+05] due to the mismatch between the real-world mobility and the classical mobility models implemented on the most popular simulators. Researchers are currently devoting lot of attention to studying real mobility models for future integration in simulators. This, together with the above considerations on the unreliability of wireless transmissions in real scenarios (especially for tiny devices), demonstrates the importance of real experimentation in the field of opportunistic networks.

## 2.3. Encoding-based Solutions

Encoding-based approaches are valuable to cope both with one-hop and end-to-end reliability. In particular, one-hop reliability takes advantage from the fact that source nodes send out *codes* rather than plain data and that codes are *redundant* such that some loss can be tolerated. This is true both with erasure codes and network coding. End-to-end reliability instead is much more concerned with exploitation of multiple simultaneous paths towards the destination and thus implies adoption of a dissemination-based data forwarding strategy which does not necessarily implies epidemic diffusion of data, but simply involves more than one next-hop node in data transmission. This means that:

- ✧ Data chunks from the same source node flow in parallel through different paths towards the destination. This is beneficial when some paths fail to deliver the data e.g., because of the presence of a selfish node along the way or because data gets lost during because a too brief contact time. This also results in an overall decrease in the delay experience of the data sent.

- ✧ By sending more data chunks than really needed data loss is better tolerated. No matter if some chunks are lost, no matter which chunks are lost, the only important thing is that enough data chunks arrive at destination.

Obviously, the more redundancy is added to the original data, the more inefficient the system may become due to the increased traffic *overhead* which implies a more consistent bandwidth consumption. In addition, memory occupancy at the relay nodes increases too.

As discussed in chapter V, different types of erasure codes have been conceived. Reed-Solomon codes are probably the most famous even though different codes which have been introduced after them, e.g., Tornado , Luby-Transform, and Raptor codes, are more efficient. The main drawback of Reed-Solomon codes is their high computational burden whereas their strength is the decoding efficiency since they require the minimum number of arrivals for the destination to be able to do the decoding. Tornado, LT, and Raptor codes have all lighter computational burden over Reed-Solomon codes, but have also lower decoding efficiency and require a higher amount of data to arrive to destination for the decoding. In sequence, Tornado, LT and Raptor codes have ever higher decoding efficiency. However, usage of Tornado, Luby-Transform and Raptor codes is limited in the research community as well as in other research and development centres because their inventors have founded a company and taken patents on them. Patents cover both the fundamental theory and specific implementation designs of these codes. Therefore, despite their higher complexity (quadratic vs. linear), Reed-Solomon codes are still valuable erasure codes. This is also proved by the fact that smart implementations of Reed-Solomon codes have been produced in the past and have been used on resource-poor systems such as obsolete processors and wireless sensor networks. Therefore, hereafter we will concentrate on Reed-Solomon codes as representative of erasure codes. Network Coding is an evolution of erasure codes and involves different stages of encoding since it is performed not only at the source nodes but also at intermediate nodes. Figure VI.1 shows the main differences and similarities between Reed-Solomon codes and Network Coding. Reed-Solomon codes have higher memory requirements at the source and destination nodes with respect to Network Coding because they exploit a pri-

| | Encoding at... | Decoding at... | Sketch factor | Permanent memory requirement at... | Packet overhead |
|---|---|---|---|---|---|
| **Reed-Solomon Codes** | SRC | DST | Fixed | SRC, DST (encoding matrix) | Corresponding row-ID in the encoding matrix |
| **Network Coding** | SRC, Relays | DST | Variable | - | Encoding Vector + Generation-ID |
| | **Loss Tolerance** | **Load Balancing** | **Robustness against node or connection failure** | **Bandwidth Occupancy** | **Computational burden at ...** |
| **Reed-Solomon Codes** | Limited by the sketch factor | NO (depends on the routing) | NO (losses exceed the max limit n-k) | Limited by the sketch factor | SRC and DST - Quadratic - |
| **Network Coding** | YES | YES | YES | Depends on • forwarding strategy • redundancy added at relays ~ network capacity | SRC, DST, and Relays - higher at DST due to gaussian elimination (limited by the generation size) - |

Figure VI.1.: Reed-Solomon Code and Network Coding: differences and similarities.

ori knowledge of the encoding matrix which must thus be permanently stored by the communicating peers. On the other hand, Network Coding does not require storage of the encoding matrix. Rather, encoding vectors must be sent along with the encoded data. As a result, Network Coding has lower memory requirements than Reed-Solomon codes but higher packet overhead that causes higher bandwidth consumption. The computational burden of Reed-Solomon codes is high at both the source and the destination nodes. In Network Coding instead, though some computation is also performed at intermediate nodes, the heaviest load is at the destination. However, this can be fixed out by managing generations. Network Coding allows load balancing and gives to the network much more robustness against node and connection failures with respect to Reed-Solomon codes. Finally, it is a better implementation of digital fountains with respect to Reed-Solomon codes because it is not constrained by a fixed sketch factor $(n/k)$, as Reed-Solomon codes are, and is thus more flexible to the changing networking conditions.

In the reference scenario discussed in section 2, communications from sensor nodes to data MULEs can greatly benefit from adoption of Reed-Solomon codes. Furthermore, Reed-Solomon codes can also be used to support opportunistic communications among data MULEs, in which case they can provide considerable added value to the data forwarding strategy applied. Better results are obviously expected with dissemination-based data forwarding strategies where multiple paths are attempted in parallel towards the destination node. Given the theoretical results which have already been presented in chapter V, application of Network Coding can surely add much more efficiency to data transmission in our reference architecture. Indeed, since with Network Coding the computational burden of source nodes is lower with respect to the case of Reed-Solomon encoding, it seems to be a more suitable approach. Nonetheless, Network Coding is more bandwidth-hungry with respect to Reed-Solomon codes (higher packet overhead) and thus when applying Network Coding, much attention should be paid to the dissemination criteria in order to avoid network congestion. Another critical issue when applying network coding is understanding the benefits on exploitation of the overall network capacity. In fact, analitycal results stating that network coding allows optimizing the bandwidth consumption up to the theoretical limit given by the *max-flow min-cut* theorem, are related to a wired Internet scenario and to networks with a fixed topology. Analogous results for wireless networks have not been obtained yet, indeed, in opportunistic scenarios with continuously changing network topologies the assumptions those results rely on are definitely not fulfilled. hence, it is not yet clear whether performance gains with network coding are sufficiently high to justify the overhead that it adds to all the participating nodes. For all the above reasons, the investigation of a network-coding-based approach to data communication in opportunistic environments is quite a complex task, and surely an interesting and challenging second step after a mature experience with erasure codes, namely Reed-Solomon codes.

The remainder of this chapter is devoted to presenting a preliminary experimental study on application of Reed-Solomon codes to transmissions from sensor nodes to data MULEs in the reference architecture. We leave investigation on network coding as future task.

# 3. Preliminary Results from a Small-scale Testbed

We are currently conducting experimentations on a small-scale testbed composed of *Mica2 mote* sensor nodes (http://www.xbow.com) running the *TinyOS*(v.1-1-14) operating system. Our aim is to compare the transmission pattern resulting from adoption of Reed-Solomon codes with that resulting from classical retransmission based approaches like *stop-and-wait* and *go-back-n*. What we especially strive to obtain is a measure of the degree of tolerance against packet loss that application of Reed-Solomon codes allows over retransmission based approach. Furthermore, we want to evaluate the improvement in the delay experienced from arrival of the first message of a burst to the last one, also in presence of data loss.

## 3.1. Testbed Description

Mica2 motes (Fig. VI.2) are small-sized sensor nodes which use an *8*-bit Atmel Microprocessor running at $4$ MHz frequency. The overall memory they use comprises $128$ KByte Instruction memory, $4$ KByte Data memory, and $512$ KByte external non-volatile storage. Radio communications are performed with an RFM ChipCom Radio running at $39.2$ Kbits/second. Some of the main features of Mica2 motes are summarized in Table VI.1. Programming Mica2 motes is accomplished via an external *Programming Board* (Fig. VI.3) which provides both a serial interface and a parallel port programming interface. During the programming phase, the programming board must be plugged into the parallel port of the computer, and the mote to be programmed must be plugged onto the programming board. Then the application is compiled on the pc and contemporary flashed to the mote through the parallel port. The serial interface of the programming board is used instead to read the data from the motes onto the pc. A mote can be programmed to send out the data both to the radio interface and to the serial interface. In both cases the output data from the mote can be read on the pc. It is necessary that the serial interface of the programming board is connected to the serial port of the pc and that the mote is plugged onto the programming board. The pc has to run an appropriate java tool which *Listens* from the serial port. If the mote is running

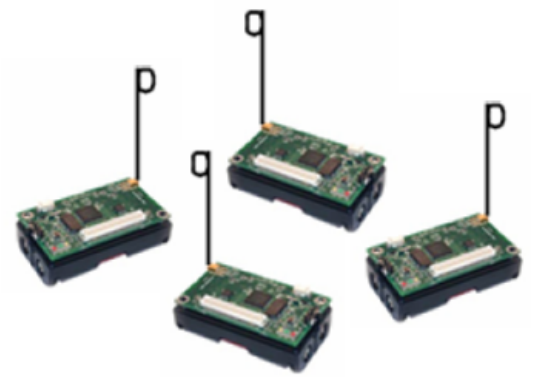

Figure VI.2.: Mica2 motes.

Table VI.1.: Mica2Motes features.

| **Processor Performance:** | |
|---|---|
| Microcontroller | ATmega128 |
| CPU Clock | 8 MHz |
| Program Flash Memory | 128 Kbytes |
| RAM | 4 Kbytes |
| Measurement (Serial) Flash | 512 Kbytes |
| Serial Communication | UART |
| **MultiChannel Radio:** | |
| Radio | ChipCom CC1000 |
| Center Frequency | 868/916 MHz |
| Data Rate | 38.4 Kbaud |
| RF Power | $-20$ to $+5$ dBm |

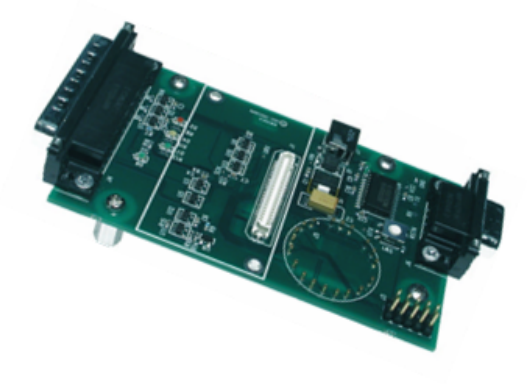

Figure VI.3.: MIB500CA Programming Board.

an application that sends data out to the serial interface then this is automatically captured on the pc. If the data which we want to capture is sent out through the radio interface, then the mote plugged onto the programming board should run an appropriate sniffer application that captures data through the radio interface and redirects it to the serial interface of the programming board (the mote sending the data to the radio interface is located apart from the programming board).

The operating system that we are currently using is TinyOS (http://www.tinyos.net/). It is currently the most famous and mature operating system for wireless sensor network programming. It has been developed at UC Berkeley and it has an open source license.

## 3.2. Experimental Results

The experimentation that we are currently conducting on the testbed described above is still in its first phase. We are testing a software prototype that implements Reed-Solomon codes. After generation of a burst of packets, the source node encodes them and produces a wider burst of coded packets, i.e., *codes*. Codes have the same size of the original packets, i.e., 20 bytes[2]. Codes are finally sent by the

[2] Please note that the size of the packets is upper-limited by the size of the payload of a TinyOS packet which is 29 bytes.

source node. The destination node waits for incoming codes and, after having received a *sufficient* number of them, performs decoding. Both source and destination nodes are aware of the encoding parameters, namely the total number of original packets and the encoding matrix. The destination node receives, together with each single code, the position of the encoding matrix that code corresponds to. For encoding and decoding, our software prototype makes use of the erasure-code functions developed at Berkeley ([KFC04]) and included in the TinyOS CVS (tinyos-1.x/contrib/GGB/apps/ErasureCode). Whenever a source sensor node has to send messages throughout the network, it bunches them together into small groups and performs encoding over each single group. The redundancy added is quite high since, for each single group, the number of codes generated is five times the number of original messages. Specifically, for each set of $7$ messages generated, the source node produces $35$ codes. The source node sends out the codes generated. The receiver node is able to decode the messages as soon as it receives a number of encoded messages corresponding to the number of the original messages generated at the source node (seven). We are testing this software prototype with only two Mica2 sensor nodes, one acting as source node and the other one as destination node. Under ideal conditions, when sensors are at a distance of $1$ m from each other and no packet loss is experienced, the decoding is very fast since it happens as soon as the first seven packets are received.

The retransmission-based scheme that we have implemented so far and that we are currently comparing with the erasure-coding-based scheme is a naive stop-and-wait scheme. The source node has a burst of $7$ packets (for fair comparison) to send out. After sending each single packet the sender waits for acknowledgement from the receiver, and sends a subsequent packet as soon as it receives the acknowledgement. To avoid infinite waits, the sender sets a timer with an appropriate *timeout* value as soon as it has sent out a packet. If the timer goes off and no acknowledgement has arrived then the sender retransmits the previous packet. The choice of an appropriate value for the timeout is critical. A too small timeout value would result in useless retransmissions whereas a too long timeout value would lengthen the total burst transmission time and lead to inefficiency. According to a first set of trials, the round-trip-time generally experienced by packets ranges between $43.60$ msec and $68.27$ msec thus, a good reference value for the

timeout to set can reasonably be 75 msec. With this timeout value we have observed the total burst transmission time when no packet loss is experienced and also in presence of limited packet loss: 20% and 40% respectively. Under ideal conditions, i.e., in absence of packet loss, the burst needs 383.82ms to reach destination, with 20% packet loss it experiences 674.25msec transmission time, and with 40% packet loss it experiences 1280.30msec transmission time. When using erasure codes the total burst transmission time also includes the time for encoding and decoding. This results in a fixed overhead which is especially disadvantageous when channel conditions are ideal and the transmission requires very short time. Whereas, when there is some packet loss and the transmission time tends to lengthen, adoption of erasure coding is quite beneficial. We observed that in absence of packet loss, the same burst of packets previously delivered with the stop-and-wait scheme requires 453.51ms delivery time with erasure coding. With 20% packet loss it needs 514.80 ms, while with 40% packet loss it needs 602.87 ms time for delivery. As is clear, in absence of packet loss, the stop-and-wait scheme performs better than the erasure-coding scheme because of the overhead introduced by encoding and decoding. However, adoption of the erasure-coding scheme is quite beneficial in presence of packet loss when it allows faster transmissions. With 20% packet loss the erasure-coding scheme allows saving 24% of the transmission time spent in case of stop-and-wait, while with 40% packet loss it allows saving 53% of the overall transmission time. This is quite promising for mobile scenarios where the packet loss experienced by packets can be far higher than 40%.

Given the redundancy of five times the number of the original packets, the software that we are testing has the potential to tolerate high packet loss, and to provide lower delay with respect to the stop-and-wait approach. This is obviously beneficial when a few amount of data is to be sent to destination. However, it is slightly inefficient when the source has a long data set (divided into different bursts) to transmit. In fact, the transmission of all the codes of a given burst occupies the channel for a long time and increases the inter-arrival time among consecutive bursts at the destination. Depending on the inter-arrival time among bursts of packets, the benefits of avoiding retransmission could be vanished. In the next experimentation phase we plan to better evaluate the benefits given by

the erasure-coding scheme over the naive stop-and-wait retransmission scheme and to define the upper limit up to which erasure coding is preferable to stop-and-wait. We also plan to compare erasure codes to a smarter retransmission-based approach, e.g., go-back-n. Finally, we intend to fine-tune the encoding parameters and the redundancy factor (ratio between the number of codes and the number of original data) to improve the performance of erasure encoding in a realistic transmission pattern in presence of node mobility.

# VII. Conclusions

Following the evolving scenario from mobile to pervasive systems, this thesis reviews different kinds of wireless environments which have been deploying over the last two decades. Starting from infrastructure-based *wireless LANs*, going further to infrastructureless *ad hoc networks* and finally to *opportunistic networks*, different networking approaches suit different user requirements and suffer from different issues some of which are still open.

Infrastructure-based WLANs are generally used to provide Internet access such that their framework is generally referred to as *wireless Internet*. They offer reliable transmission of data, thank to the benefits of the infrastructure, however they tend to be used to provide the same kinds of services of the wired Internet which are quite resource-hungry. This causes the need to optimize service provisioning so as to mask the presence of the last wireless hop and guarantee a seamless wired-wireless infrastructure towards the mobile user.

In the *wireless Internet* scenario, we have focused on energy efficiency issues and discussed a software solution for management of the power-consuming states of the wireless network interface during service provisioning. We have particularly targeted services which are characterised by strict quality-of-service requirements, i.e., streaming services. When dealing with these kinds of services reliable transmission of stream chunks is not sufficient, instead *timely* delivery is required, hence, providing energy saving support is a real challenge because it concerns with putting the wireless interface card into sleep mode when not strictly necessary. The solution presented fulfils both energy-saving and quality-of-service requirements of real-time streaming services.

When passing from infrastructure-based to infrastructure-less scenarios, and particularly to *Multi-hop Ad hoc NETworks*, one of the major issues arising is con-

cerned with the absence of a real deployment. Though great volume of work has been conducted so far on MANETs via analitical and simulation studies, much work still remains to be done on real testbeds. Lessons learned from a real experience on small-scale MANETs consisting of $8$ to $12$ nodes (both laptops and iPAQs) highlight the need to better tailor user services to the MANET architecture. In fact, when porting *as they are* some of the most popular applications of the Internet, e.g., p2p applications, to a MANET environment, clear inefficiencies arise. Not only is the quality of service perceived by a MANET user far lower than the quality of service perceived by a wired-Internet user or even by a wireless-Internet user, but the system sometimes is not able to provide the service at all.

Going further on to the investigation of an ever weaker wireless infrastructure, *opportunistic networks* offer a completely new concept of wireless networks. They do not rely on any fixed infrastructure, nor try to self-configure a temporary infrastructure over nodes in proximity of each other. Opportunistic networks instead, rely on *contact opportunities* that may arise between nodes carried by users involved in their everyday activities (e.g., at work, on buses, at school/university, etc.) Messages are exchanged whenever and wherever possible and their delivery success is deeply tied with social dynamics and history of encounters among people. Given the *extremely high* mobility that characterizes these emerging wireless frameworks, added to the well-known *lossy* nature of the communication channel, *unreliability* of transmissions arises as a major compelling issue. In fact, communications can only take place during contact times and must be extremely fast and effective. A valuable strategy to cope with unreliability issues in so-called *extreme environments*, is the combined action of *encoding techniques* and *dissemination*-based data forwarding strategies. This approach relies on the generation of redundancy in both data and paths. Redundancy of data gives much robustness against data loss since it is required that only a subset of the codes generated arrives at destination to allow correct reconstruction of the original information. Redundancy of paths instead, is necessary because it is not possible to predict in advance the evolving sequence of contacts that may bring the data to destination and hence, giving custody of the information to a single next hop (i.e., relying on a single path) is extremely unsafe.

Opportunistic networks characterized by low-powered resource-constrained de-

vices offer currently the framework that best matches the concept of pervasive systems. In this last scenario, we are focusing on the case of *sparse Wireless Sensor Networks* where sensors are deployed in the environment for monitoring purposes and the data that they collect is read by mobile agents passing by, named *data MULEs*. Data MULEs can utilize the information read at the sensor nodes to run context-aware applications or simply forward it until reaching destination *warehouses* for final processing and storage. This is completely coherent with the vision of a pervasive environment where devices are embedded in the environment and interact continuously and transparently with users who live in within or simply pass by. The interaction between embedded sensor nodes and data MULESs is only the first step of the overall communication pattern. Data MULEs are expected to exchange each other the information they carry in an opportunistic fashion. Eventually, data may reach infrastructure access points from which it can be delivered to the farthest destinations. In this environment, wireless communications naturally complete interactions among users and occur whenever/wherever user meet up with each other or get accidentally in proximity of each other. Depending on the particular application which is deployed, wireless communications can also be used to create *social opportunities* in places where human interactions do not generally occur (e.g., on buses or in the underground while moving from place to place). To support a similar framework it is necessary to develop new communication paradigms which take into account the absence of stable links between communication peers (i.e., *intermittent* connectivity) and rather expect the occurrence of brief contact times. Indeed, the new communication paradigms should consider the absence of a complete known path from source to destination nodes and rather exploit forms of data forwarding which are more similar to the way social communcations/exchanges occur. We believe that encoding-based data forwarding strategies offer a valuable solution to support the emerging pervasive framework.

We are currently conducting an experimental study on sparse wireless sensor networks aimed at investigating the first step of this communication pattern. Particularly, we are focusing on communications from sensor nodes to mobile data MULEs. After the preliminary set-up phase of a small-scale testbed composed of wireless sensor nodes, we are currently analysing the efficacy of encoding-based

transmission schemes with respect to classical retransmission-based schemes. Preliminary results demonstrate the feasibility of this solution and show a degree of tolerance to data loss which is five times that of a naive retransmission based scheme (i.e., stop and wait). We plan to contrast encoding transmission schemes with cleverer retransmission-based schemes so as to have a complete comparison which includes both the better and the worst case. Our main target is to define potentialities and limits of this approach and to understand if there are drawbacks in their application. We intend to measure the degree of robustness against data loss and also to identify possible integrations between encoding- and retransmission-based schemes.

As further step, we plan to extend application of encoding-based schemes to communications among data MULEs in a fully opportunistic network. In this case it will be possible to use more complex encoding systems since communications between data MULEs can exploit powerful devices with less constraints on both memory space and computational power. Among the major concerns of this further investigation will be the *integration* between encoding schemes and data forwarding strategies, and the choice of a valuable *evaluation method*. This should take into account realistic *mobility models* which well-suit social interactions among individuals and groups. As much as the mobility model reflects real users' behavior, it can be considered eligible to determine either the success or failure of the proposed communication paradigm.

# Bibliography


[AAC+03] I. F. Akyildiz, O. Akan, C. Chen, J. Fang, and W. Su. Interplanetary internet: state-of-the-art and research challenges. *Computer Networks*, 43(2):75–112, 2003. 71

[ABCG04] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Wi-Fi in Ad Hoc Mode: A Measurement Study. In *Proceedings of PerCom 2004*, Orlando, Florida, March 2004. 42, 43, 44

[ACG04a] G. Anastasi, M. Conti, and E. Gregori. *Mobile Ad Hoc Networking*, chapter "IEEE 802.11 Ad Hoc Networks: Protocols, Performance, and Open Issues of Ad Hoc Networks". IEEE Press and John Wiley and Sons, Inc., New York, 2004. S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic (Editors). 44

[ACG+04b] G Anastasi, M. Conti, E. Gregori, A. Falchi, and A. Passarella. Performance measurements of mote sensor networks. In *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile System*, Venice, Italy, October 2004. 201

[ACG+05a] G. Anastasi, M. Conti, E. Gregori, A. Passarella, and L. Pelusi. A Power-Aware Multimedia Streaming Protocol for Mobile Users. In *Proceedings of the IEEE International Conference on Pervasive Services (ICPS '05)*, Santorini, July 2005. 6

[ACG+05b] G. Anastasi, M. Conti, E. Gregori, A. Passarella, and L. Pelusi. A Power-Aware Multimedia Streaming Protocol for Mobile Users. Technical report 2005/01, IIT-CNR, February 2005. http://bruno1.iit.cnr.it/ bruno/techreport.html. 18, 27

[ACG+06a] G. Anastasi, M. Conti, E. Gregori, A. Passarella, and L. Pelusi. An Energy-efficient Protocol for Multimedia Streaming in a Mobile Environment. *Journal of Pervasive Computing and Communications*, January 2006. (Troubador Publishing). 6

[ACG+06b] G. Anastasi, M. Conti, E. Gregori, C. Spagoni, and G. Valente. Motes sensor networks in dynamic scenarios: an experimental study for pervasive applications in urban environments. *International Journal of Ubiquitous Computing and Intelligence*, 1(1), January 2006. 8, 202

[ACGP03a] G. Anastasi, M. Conti, E. Gregori, and A. Passarella. Balancing Energy Saving and QoS in the Mobile Internet: An Application-Independent Approach. In *Proceedings of the* $36^{th}$ *Hawaii International Conference on System Sciences (HICSS-36)*, Hawaii, January 2003. 14, 28

[ACGP03b] G. Anastasi, M. Conti, E. Gregori, and A. Passarella. Performance Comparison of Power Saving Strategies for Mobile Web Access. *Performance Evaluation Journal*, 53(3–4):273–294, 2003. 14, 28

[ACL00] G. Anastasi, M. Conti, and W. Lapenna. Power Saving Policies for Wireless Access to TCP/IP Networks. In *Proceedings of the* $8^{th}$ *IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks (IFIP ATM&IP2000)*, Ilkley, UK, July 2000. 13, 14

[ACLY00] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, July 2000. http://personal.ie.cuhk.edu.hk/ pwkwok4/Yeung/1.pdf. 179, 181

[ACP05] G. Anastasi, M. Conti, and A. Passarella. *Algorithms and Protocols for Wireless and Mobile Networks*, chapter "Power Management in Mobile and Pervasive Computing Systems". CRC_Hall, 2005. 11

[ANF05] M. Anand, E. B. Nightingale, and J. Flinn. Self-Tuning Wireless Network Power Management. *Wireless Networks*, 11(4), 2005. 13, 14

[AS98] S. Acharya and B. C. Smith. Compressed Domain Transcoding of MPEG. In *Proceedings of IEEE Conference on Multimedia Computing Systems (ICMCS)*, Austin, TX, 1998. 14

[AVAS06] I. F. Akyildiz, M. C. Vuran, O. B. Akan, and W. Su. Wireless Sensor Networks: A Survey Revisited. *Computer Networks*, 2006. 147

[BB97] A. Bakre and B. R. Badrinath. Implementation and Performance Evaluation of Indirect TCP. *IEEE Transactions on Computers*, 46(3), March 1997. 13

[BB04] A. Boukerche and L. Bononi. *Mobile Ad Hoc Networking*, chapter "Simulation and Modeling of Wireless, Mobile, and Ad Hoc networks". IEEE Press and John Wiley and Sons, Inc., New York, 2004. S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic (Editors). 42

[BBL05] B. Burns, O. Brock, and B. N. Levine. MV Routing and capacity building in disruption tolerant networks. In *Proceedings of the IEEE INFOCOM 2005*, Miami, FL, March 2005. 9, 105, 106, 203

[BC00] N. Bjork and C. Christopoulos. Video transcoding for universal multimedia access. In *Proceedings of the* $8^{th}$ *ACM Conference on Multimedia*, Los Angeles, CA, November 2000. 14

[BCDP04] E. Borgia, M. Conti, F. Delmastro, and L. Pelusi. Lessons from an ad hoc network test-bed: middleware and routing issues. *Ad Hoc & Sensor Wireless Networks: An International Journal*, 1(1):125–157, January 2004. (Old City Publishing). 6

[BCG04] *Mobile Ad Hoc Networking*. IEEE Press and John Wiley and Sons, Inc., New York, 2004. Basagni, S. and Conti, M. and Giordano, S. and Stojmenovic, I. (Editor). 41, 53

[BCG05] R. Bruno, M. Conti, and E. Gregori. Mesh Networks: Commodity Multihop Ad Hoc Networks. *IEEE Communications Magazine*, pages 123–133, March 2005. 147

[BEH03] J. Blum, A. Eskandarian, and L. J. Hoffman. Performance Characteristics of Inter-Vehicle Ad Hoc Networks. In *Proceedings of the 6th IEEE International Conference on Intelligent Transportation Systems*, Shanghay, China, 2003. 87

[Bis01] C. Bisdikian. An Overview of the Bluetooth Wireless Technology. *IEEE Communications Magazine*, December 2001. 44

[BLB02] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA02)*, June 2002. 138

[BLM02] J. B. Byers, M. Luby, and M. Mitzenmacher. A Digital Fountain Approach to Asynchronous Reliable Multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), October 2002. 156, 165

[BM03] S. Biswas and R. Morris. Opportunistic Routing in Multihop Wireless Networks. In *HotNets workshop*, 2003. 120

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, Dallas, Texas, USA, October 1998. 42

[BMJ00] J. Broch, D. A. Maltz, and D. B. Johnson. Quantitative lessons from a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. In *Proceedings of the IEEE Wireless Communications and Network Conference (WCNC 2000)*, 2000. 42

[CBH+01] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss. Interplanetary Internet (IPN): Architectural Definition, May 2001. Internet draft - https://www.cs.tcd.ie/Stephen.Farrell/ipn/background/draft-irtf-ipnrg-arch-00.txt. 71

[CBH+02] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Network Architecture: The Evolving Interplanetary Internet, August 2002. Internet draft. 76

[CBH+03] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking: An Approach to Interplanetary Internet. *IEEE Communications Magazine*, June 2003. 71, 76

[CBH+05] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Network Architecture, July 2005. Internet draft - http://www.dtnrg.org/docs/specs/draft-irtf-dtnrg-arch-03.txt. 76

[CCL03] I. Chlamtac, M. Conti, and J. Liu. Mobile Ad hoc Networking: Imperatives and Challenges. *Ad Hoc Networks Journal*, 1(1), January–March 2003. 41, 147

[CCMT04] M. Conti, J. Crowcroft, G. Maselli, and G. Turi. *Handbook on Theoretical and Algorithmic Aspects of Sensors, Ad Hoc Wireless, and Peer-to-Peer Networks*, chapter "A Modular Cross-Layer Architecture for Ad Hoc Networks". CRC Press, New York, 2004. 68

[CFS06] C. Chekuri, C. Fragouli, and E. Soljanin. On average throughput and alphabet size in network coding. *IEEE/ACM Transactions on Networking (TON), Special Issue on Networking and Information Theory*, 14:2410–2424, June 2006. 181

[CGT04] M. Conti, E. Gregori, and G. Turi. Towards scalable P2P computing for mobile ad hoc networks. In *Proceedings of the workshop on Mobile Peer-to-Peer computing (MP2P'04), IEEE PerCom 2004*, 2004. 46

[Cha03] S. Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. *Multimedia Systems*, 9(2), August 2003. 15

[CHC+05] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report UCAM-CL-TR-617, Computer Laboratory, University of Cambridge, February 2005. 85, 109, 204

[CKV01] Z. D. Chen, HT Kung, and D. Vlah. Ad hoc Relay Wireless Networks over Moving Vehicles on Highways. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001)*, October 2001. 87, 88, 126

[CM01] X. Chen and A. L. Murphy. Enabling disconnected transitive communication in mobile ad hoc networks. In *Proceedings of the Workshop on Principles of Mobile Computing (co-located with PODC'01)*, Newport, RI, USA, August 2001. 113

[CMTG04] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross Layering in Mobile Ad Hoc Network Design. *IEEE Computer*, February 2004. 68

[CNP+01] I. Chatzigiannakis, S. Nikoletseas, N. Paspallis, P. Spirakis, and C. Zaroliagis. An experimental study of basic communication protocols in ad-hoc mobile networks. *Lecture Notes in Computer Science*, 2141:159–169, 2001. 136

[CNS01] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. Analysis and experimental evaluation of an innovative and efficient routing protocol for ad-hoc mobile networks. *Lecture Notes in Computer Science*, 1982:99–111, 2001. 136

[Con04] M. Conti. *Environments: Technologies, Protocols and Applications*, chapter "Wireless Communications and Pervasive Technologies", pages 63–99. John Wiley & Sons, 2004. 147

[Conar] M. Conti. *The Handbook of Computer Networks*, chapter "Principles and Applications of Ad Hoc and Sensor Networks". John Wiley & Sons, to appear. 147

[CV02] S. Chandra and A. Vahdat. Application-specific Network Management for Energy-aware Streaming of Popular Multimedia Formats. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, 2002. 14, 15

[D5] MobileMAN Deliverable D5. Deliverable. http://cnd.iit.cnr.it/mobileMAN/pub-deliv.html. 46, 47

[D8] MobileMAN Deliverable D8: First Phase. Deliverable. http://cnd.iit.cnr.it/mobileMAN/pub-deliv.html. 52

[DCABM03] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM/IEEE MobiCom*, September 2003. 120

[DCG+03] H. Dong, I. D. Chakeres, A. Gersho, E. M. B. R. U. Madhow, and J. D. Gibson. Speech Coding for Mobile Ad hoc Networks. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2003. 149

[DCY00] S. R. Das, R. Castaneda, and J. Yan. Simulation Based Performance Evaluation of Mobile, Ad Hoc Network Routing Protocols. *ACM/Baltzer Mobile Networks and Applications (MONET) Journal*, July 2000. 42

[DEH+05] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Médard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN)*, London, UK, May 2005. 150

[DFL01] J. A. Davis, A. H. Fagg, and B. N. Levine. Wearable Computers as Packet Transport Mechanisms in Highly-Partitioned Ad-Hoc Networks. In *Proceedings of the Inter-Symposium on Wearable Computing*, Zurich, October 2001. 103, 105, 124, 203

[DGL+04] S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvartsman, and J. L. Welch. Virtual Mobile Nodes for Mobile Ad Hoc Networks. In

*Proceedings of the 18th International Symposium on Distributed Computing (DISC)*, 2004. 137, 138

[DPR00] S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *Proceedings INFOCOM 2000*, Tel Aviv, Israel, March 2000. 42

[DPR05] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN 2005)*, Sunset Village, UCLA, Los Angeles, CA, April 2005. 149, 175, 176

[DUP02] A. Doria, M. Uden, and D. P. Pandey. Providing connectivity to the saami nomadic community. In *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation (dyd 02)*, Bangalore, India, December 2002. 77

[DZD+03] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for Structured Peer-to-Peer Overlays. In *Proceedings of the the 2nd International Workshop on Peer-to-peer Systems (IPTPS'03)*, Berkeley, CA, February 2003. 52

[ECPS02] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the Physical World with Pervasive Networks. *IEEE Pervasive Computing*, 1(1), 2002. 4

[Enk03] W. Enkelmann. FleetNet - Applications for Inter-Vehicle Communication. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2003)*, June 2003. 89

[FFH+04] A. Festag, H. Füßler, H. Hartenstein, A. Sarma, and R. Schmitz. FleetNet: Bridging car-to-car communication into the real world. In *Proceedings of the 11th World Congress on ITS*, Nagoya, Japan, October 2004. 90

[Fin04] R. Finlayson. A More Loss-Tolerant RTP Payload Format for MP3 Audio, October 2004. Internet-draft, updates RFC 3119 - http://www.cs.columbia.edu/ hgs/rtp/drafts/draft-ietf-avt-rfc3119bis-03.txt. 26

[FKV02] H. Füßler, M. Käsemann, and D. Vollmer. A Comparison of Strategies for Vehicular Ad-Hoc Networks. Technical Report TR-3-2002, Department of Computer Science, University of Mannheim, July 2002. 90

[FLBW06] C. Fragouli, J. Y. Le Boudec, and J. Widmer. Network Coding: An Instant Primer. *ACM Computer Communication Review*, 36(1):63–68, 2006. 181

[FMH+02] H. Füßler, M. Mauve, H. Hartenstein, M. Käsemann, and D. Vollmer. Location-Based Routing for Vehicular Ad-Hoc Networks. In *Proceedings of the ACM MobiCom 2002*, Atlanta, Giorgia, USA, September 2002. 90

[FR97] W. C. Feng and J. Rexford. A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video. In *Proceedings of the IEEE INFOCOM'97*, pages 58–66, April 1997. citeseer.ist.psu.edu/feng97comparison.html. 12, 19

[FRZ+05] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, May 2005. 167

[FS04] C. Fragouli and E. Soljanin. Decentralized network coding. In *Proceedings of the Information Theory Workshop*, October 2004. 189

[Gal63] R. G. Gallagar. Low Density Parity-Check Codes, 1963. Cambridge MA: MIT Press. 167

[GBMY97] D. Goodman, J. Borras, N. Mandayam, and R. Yates. INFOSTATIONS: A new system model for data and messaging services. In *Proceedings of the IEEE VTC'97*, May 1997. 132

[Gha04] L. Gharai. RTP Profile for TCP-Friendly Rate Control, August 2004. Internet-draft - http://www.east.isi.edu/ ladan/publications/tfrc-profile.txt. 17

[GK00] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46:388–404, March 2000. 79

[GLNT] P. Gunningberg, H. Lundgren, E. Nordstrom, and C. Tschudin. Lessons from Experimental MANET Research. *To appear in Ad Hoc Networks Journal, special issue on "Ad Hoc Networking for Pervasive Systems"*. M. Conti, E. Gregori (Editors). 6, 42, 43

[GR05] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of INFOCOM'05*, Miami, FL, March 2005. 193

[GT02] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4), August 2002. 79, 81

[HBT06] A. A. Hamra, C. Barakat, and T. Turletti. Network Coding for Wireless Mesh Networks: A Case Study. In *Proceedings of the 2nd IEEE International Conference on the World of Wireless, Mobile and Multimedia Networks (WoWMoM2006)*, Niagara-Falls / Buffalo, NY, June 2006. 150

[HFPW03] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP-Friendly Rate Control (TFRC): Protocol Specification, January 2003. RFC 3448 - http://www.faqs.org/rfcs/rfc3448.html. 16

[HHHK02] L. Huang, F. Hartung, U. Horn, and M. Kampmann. A Proxy-based TCP-friendly streaming over mobile networks. In *Proceedings of the* $5^{th}$ *ACM international workshop on Wireless mobile multimedia*, Atlanta, September 2002. 22

[HKA04] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of ACM Mobicom*, 2004. 85, 125

[HKM+03] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2003)*, Yokohama, Japan, July 2003. 181, 189

[HMS+03] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, October 2003. 181

[Hui03] C. Huitema. Real Time Control Protocol (RTCP): attribute in Session Description Protocol (SDP), October 2003. RFC 3605 - http://www.faqs.org/rfcs/rfc3605.html. 17

[IR00] A. L. Iacono and C. Rose. Infostations: New Perspectives on Wireless Data Networks. Technical document, WINLAB, Rutgers University, 2000. 132

[JHP+03] J. G. Jetcheva, Y. C. Hu, S. PalChaudhuri, A. K. Saha, and D. B. Johnson. Design and Evaluation of a Metropolitan Area Multitier Wireless Ad Hoc Network Architecture. In *Proceedings of the 5th IEEE International Workshop on Mobile Computing Systems & Applications*, Monterey, CA, October 2003. 92

[JLHM99] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek. Routing Protocols for Mobile Ad-Hoc Networks: A Comparative Performance Analysis. In *Proceedings of The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, Seattle, Washington, USA, August 1999. 42

[JLW05] E. P. C. Jones, L. Li, and P. A. S. Ward. Practical Routing in Delay-Tolerant Networks. In *Proceedings of the ACM SIGCOMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, USA, August 2005. 124

[JOW+02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. I. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM SIGPLAN Notices*, 37:96–107, 2002. 95, 109

[JSB+05] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks. *ACM/Kluwer Mobile Networks and Applications Journal*, 2005. 139

[KB02] R. Krashinsky and H. Balakrishnan. Minimizing Energy for Wireless Web Access with Bounded Slowdown. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom)*, 2002. 29

[KFC04] S. Kim, R. Fonseca, and D. Culler. Reliable Transfer on Wireless Sensor Networks. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004)*, October 2004. 149, 166, 167, 211

[KK98] R. Kravets and P. Krishnan. Power Management Techniques for Mobile Communication. In *Proceedings of the* $4^{th}$ *Annual ACME/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, 1998. 13, 14

[KM88] J. F. Kurose and H. Mouftah. Computer-Aided Modeling of Computer Communication Networks. *IEEE Journal on Selected Areas in Communications*, 6(1):130–145, January 1988. 41

[KM02] R. Koetter and M. Médard. Beyond Routing: An Algebraic Approach to Network Coding. In *Proceedings of INFOCOM'02*, 2002. 179

[KOX05] P. Karlsson, L. Öberg, and Y. Xu. An Address Coding Scheme for Wireless Sensor Networks. In *Proceedings of the 5th Scandinavian Workshop on Wireless Ad-hoc Networks (ADHOC '05)*, Stockholm, May 2005. 149

[KRH+06] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in The Air: Practical Wireless Network Coding. In *Proceedings of the ACM SIGCOMM 2006*, Pisa, Italy, September 2006. 150

[Lav83] S. S. Lavenberg. *Computer Performance Handbook*. Academic Press, New York, 1983. 41

[LDS03] A. Lindgren, A. Doria, and O. Schelèn. Probabilistic routing in intermittently connected networks. *Mobile Computing and Communications Review*, 7(3), July 2003. 9, 117, 203

[LFC05] J. Leguay, T. Friedman, and V. Conan. DTN Routing in a Mobility Pattern Space. In *Proceedings of the ACM SIGCOMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, USA, August 2005. 118, 119

[LHT+03] C. Lochert, H. Hartenstein, J. Tian, H. Füßler, D. Herrmann, and M. Mauve. A Routing Strategy for Vehicular Ad Hoc Networks in City Environments. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2003)*, Columbus, OH, USA, June 2003. 90

[Lin70] S. Lin. *An Introduction to Error-Correcting Codes*. Prectice-Hall, Inc., New Jersey, 1970. Englewood Cliffs (Editor). 162

[Lin82] J. H. Lint. *Introduction to the Coding Theory*. Springer-Verlag, Berlin, 1982. 162

[LL04] Z. Li and B. Li. Network coding in undirected networks. In *Proceedings of the Annual Conference on Information Sciences and Systems (CISS)*, 2004. 181

[LMS+97] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical Loss-Resilient Codes. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 150–159, 1997. 167

[LMSS01] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, February 2001. 167, 169, 170, 171

[LNT02] H. Lundgren, E. Nordstrom, and C. Tschudin. Coping with Communication Gray Zones in IEEE 802.11 based Ad Hoc Networks. In *Proceedings of WoWMoM 2002*, Atlanta, GA, September 2002. 42

[LPY+06] U. Lee, J. S. Park, J. Yeh, G. Pau, and M. Gerla. CodeTorrent: Content Distribution using Network Coding in VANETs. In *Proceedings of the 1st International ACM Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (ACM MobiShare), in conjunction with ACM Mobicom 2006*, Los Angeles, CA, USA, September 2006. 150

[LR00] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobicom)*, Boston, August 2000. 133

[Lub02] M. Luby. LT codes. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 271–282, 2002. 172, 174

[LYC03] S. Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003. 181

[MC04] J. P. Macker and S. Corson. *Mobile Ad hoc networking*, chapter "Mobile Ad hoc Networks (MANET): Routing technology for dynamic, wireless networking". IEEE Press and John Wiley & Sons, Inc., New York, 2004. 147

[MFHF04] M. Möske, H. Füßler, H. Hartenstein, and W. Franz. Performance Measurements of a Vehicular Ad Hoc Network. In *Proceedings of the IEEE Vehicular Technology Conference (VTC'04 Spring)*, Milan, Italy, May 2004. 90

[MHM05] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *Proceedings of the 6th IEEE International Symposium on a World of Wireless, Mobile and*

*Multimedia Networks (WoWMoM 2005)*, Taormina-Giardini Naxos, Italy, June 2005. 9, 126, 203

[MV04] M. McNett and G. M. Voelker. Access and mobility of wireless PDA users. Technical report, Computer Science and Engineering, UC San Diego, 2004. 85

[NMY03] T. Noguchi, T. Matsuda, and M. Yamamoto. Performance Evaluation of New Multicast Architecture with Network Coding. *IEICE Trans. Comm.*, June 2003. 181

[NPB03] D. Nain, N. Petigara, and H. Balakrishnan. Integrated Routing and Storage for Messaging Applications in Mobile Ad Hoc Networks. In *Proceedings of WiOpt*, Autiplis, France, March 2003. 130

[OZRM00] L. E. Owen, Y. Zhang, L. Rao, and G. McHale. Traffic Flow Simulation Using CORSIM. In *Proceedings of the 2000 Winter Simulation Conference*, 2000. 87

[PFH02] A. Pentland, R. Fletcher, and A. A. Hasson. A road to universal broadband connectivity. In *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation (dyd 02)*, Bangalore, India, December 2002. 144

[PFH04] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *IEEE Computer*, 37(1):78–83, January 2004. 144

[PPC] L. Pelusi, A. Passarella, and M. Conti. *Handbook of Wireless Ad hoc and Sensor Networks*, chapter "Encoding for Efficient Data Distribution in Multi-hop Ad hoc Networks". Wiley and Sons. A. Boukerche (Editor), to appear. 9

[PPC06a] L. Pelusi, A. Passarella, and M. Conti. Beyond MANETs: dissertation on Opportunistic Networking. Technical report 2006/01, IIT-CNR, Computer Networks Department, May 2006. http://bruno1.iit.cnr.it/ bruno/techreport.html. 8

[PPC06b] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad hoc Networks. *IEEE Communications Magazine*, 44(11), November 2006. 8, 148

[PR99] C. E. Perkins and E. M. Royer. Ad hoc On Demand Distance Vector Routing. In *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999. 45

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object localtion and routing for large scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001. 47, 54

[Riz97a] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. *ACM Computer Communication Review*, 27(2):24–36, April 1997. 155, 164, 166

[Riz97b] L. Rizzo. On the feasibility of software FEC. Technical Report LR-970131, DEIT, January 1997. http://www.iet.unipi.it/ luigi/softfec.ps. 155, 164, 166

[Rob02] S. Robinson. Beyond Reed-Solomon: New Codes for Internet Multicasting Drive Silicon Valley Start-up, May 2002. SIAM News - Volume 35, Number 4. 175

[Roy04] E. B. Royer. *Mobile Ad Hoc Networking*, chapter "Routing approaches in Mobile Ad Hoc Networks". IEEE Press and John Wiley and Sons, Inc., New York, 2004. S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic (Editors). 41, 45

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, June 1960. 157, 158

[RSU00] T. Richardson, M. A. Shokrollahi, and R. Urbanke. Design of capacity approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47:619–637, February 2000. 174

[RV98] L. Rizzo and L. Vicisano. RMDP: an FEC-based Reliable Multicast protocol for wireless environments. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2(2):23–31, 1998. 149, 155, 164, 166

[Sat02] M. Satyanarayanan. A catalyst for mobile and ubiquitous computing. *IEEE Pervasive Computing*, 1(1), January-March 2002. 3

[SBRJ02] C. Shen, G. Borkar, S. Rajagopalan, and C. Jaikaeo. Interrogation-Based Relay routing for Ad hoc Satellite networks. In *Proceedings of IEEE Globecom 02*, Taipei, Taiwan, November 2002. 125

[SBS+03] J. P. Singh, N. Bambos, B. Srinivasan, D. Clawin, and Y. Yan. Proposal and demonstration of link connectivity assessment based enhancements to routing in mobile ad-hoc networks. In *Proceedings of the IEEE Vehicular Technology Conference*, Fall 2003. 128

[SBSC02] J. P. Singh, N. Bambos, B. Srinivasan, and D. Clawin. Wireless LAN Performance under Varied Stress Conditions in Vehicular Traffic Scenarios. In *Proceedings of the IEEE Vehicular Technology Conference*, Vancouver, Canada, Fall 2002. 91

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, March 2003. Internet-draft, updates RFC 1889 - http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-new-12.ps. 16

[Sch99] H. Schulzrinne. RTP site, 1999. http://www.cs.columbia.edu/ hgs/rtp/. 16

[Sch02] S. Schultz. Engineers and biologists design wireless devices to unlock secrets of animal kingdom, November 2002. http://www.princeton.edu/pr/pwb/02/1111/. 94, 109

[SCP+04] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User Mobility for Opportunistic Ad-Hoc Networking. In *Proceedings of the 6th IEEE Workshop on Mobile Computing System and Applications (WMCSA)*, UK, December 2004. 84

[SET03] P. Sanders, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In *Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures*, 2003. 181, 189

[SFP04] J. Sushant, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of SIGCOMM'04*, August 2004. 98, 99, 124, 125

[SH03] T. Small and Z. J. Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way). In *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, Annapolis, MD, USA, June 2003. 96, 133

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, July 1948. 150

[Sho04] A. Shokrollahi. Raptor codes. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2004)*, Chicago, IL, USA, July 2004. Preprint available at http://algo.epfl.ch/pubs/raptor.pdf. 176

[SK96] M. Stemm and R. H. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices. In *Proceedings of the $3^{rd}$ International Workshop on Mobile Multimedia Communication*, Princeton, NJ, September 1996. 14

[SLL06] A. Shokrollahi, S. Lassen, and M. Luby. Multi-stage code generator and decoder for communication systems, June 2006. U.S. Patent Application #20030058958. 157

[SM04a] G. Sharma and R. R. Mazumdar. Delay and Capacity Trade-off in Wireless Ad Hoc Networks with Random Mobility. Preprint, School of ECE, Purdue University, 2004. http://www.ece.purdue.edu/ mazum/MONET2004.pdf. 81

[SM04b] G. Sharma and R. R. Mazumdar. On Achievable Delay/Capacity Trade-offs in Mobile Ad Hoc Networks. In *Proceedings of the Work-*

*shop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WIOPT)*, Cambridge, UK, March 2004. 81

[SM04c] G. Sharma and R. R. Mazumdar. Scaling laws for capacity and delay in wireless ad hoc networks with random mobility. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Paris, France, June 2004. 81

[SPR05] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proceedings of the ACM SIGCOMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, USA, August 2005. 107, 108

[SR03] P. Shenoy and P. Radkov. Proxy-Assisted Power-Friendly Streaming to Mobile Devices. *Multimedia Computing and Networking*, 2003. 15, 16

[SRJB03] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In *IEEE SNPA Workshop*, May 2003. 139, 198

[SRL04] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP), February 2004. Internet-draft, updates RFC 2326 - http://www.rtsp.org/2004/drafts/draft06/draft-ietf-mmusic-rfc2326bis-06.txt. 16

[Ton] A. Tonnesen. OLSR: Optimized Link State Routing protocol. http://www.olsr.org. 45, 51

[TR01] F. Tchakountio and R. Ramanathan. Tracking highly mobile endpoints. In *ACM Workshop on Wireless Mobile Multimedia (WoWMoM)*, Rome, Italy, July 2001. 129

[TZZ03] K. Tan, Q. Zhang, and W. Zhu. Shortest path routing in partially connected ad hoc networks. In *IEEE Globecom*, 2003. 122

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Department of Computer Science, Duke University, Durham, NC, 2000. 9, 101, 105, 203

[VKR+05] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, November 2005. 141

[War03] F. Warthman. Delay-Tolerant Networks (DTNs) - A tutorial v1.1, March 2003. http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf. 76

[WCK05a] Y. Wu, P. A. Chou, and S. Y. Kung. Information Exchange in Wireless Networks with Network Coding and Physical-layer Broadcast. In *Proceedings of the 39th Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, March 2005. 150

[WCK05b] Y. Wu, P. A. Chou, and S. Y. Kung. Minimum-Energy Multicast in Mobile Ad Hoc Networks using Network Coding. *IEEE Transactions on Communications*, 53(11):1906–1918, November 2005. 150

[Wei91] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, September 1991. 3

[WFLB05] J. Widmer, C. Fragouli, and J. Y. Le Boudec. Low-complexity energy-efficient broadcasting in wireless ad hoc networks using network coding. In *Proceedings of the 1st Workshop on Network Coding, Theory, and Applications*, April 2005. 150

[WJMF05] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure Coding Based Routing for Opportunistic Networks. In *in Proceedings of the ACM SIGCOMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, August 2005. 108

[WLB05] J. Widmer and J. Y. Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *Proceedings of the ACM SIG-*

*COMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, USA, August 2005. 109, 111, 150, 192

[WSa] The FleetNet Project web-site. http://www.et2.tu-harburg.de/fleetnet/index.html. 89

[WSb] Avalanche: File swarming with network coding. http://research.microsoft.com/pablo/avalanche.aspx. 193

[WSc] The ns-2 network simulator. http://www.isi.edu/nsnam/ns/. 87

[WSd04] The ZebraNet Wildlife Tracker, January 2004. http://www.princeton.edu/ mrm/zebranet.html. 94, 109

[WSe] MPALA Wildlife Foundation. http://www.mpala.org/researchctr/index.html. 94, 109

[WSf] IEEE 802.11 WLAN. http://grouper.ieee.org/groups/802/11/main.html. 12, 43

[WSh] FreePastry Website. http://freepastry.rice.edu. 47, 52

[WSi] APE: Ad hoc Protocol Evaluation testbed. http://apetestbed.sourceforge.net/. 42

[WSm] AODV: Ad hoc On demand Distance Vector routing. http://user.it.uu.se/ henrikl/aodv/. 51

[WSn] The official Bluetooth web site. http://www.bluetooth.com/. 44

[WSo] Java API specification. http://java.sun.com. 52

[ZAZ04] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing (Mobihoc)*, May 2004. 142

[ZD04] G. Zaruba and S. Das. *Mobile Ad Hoc Networking*, chapter "Off-the-Shelf Enablers of Ad Hoc Networks". IEEE Press and John Wiley and Sons, Inc., New York, 2004. S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic (Editors). 44

[ZH97] J. Zhang and J. Y. Hui. Traffic characteristics and smoothness criteria in VBR video traffic smoothing. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1997. 21

[Zha96] J. Zhang. Optimal buffering algorithms for client-server VBR video retrievals. PhD thesis, Rutgers University, 1996. 19, 21, 22, 23, 24

[ZHKS04] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of MobiSys'04*, Boston, MA, June 2004. 201