Delay Tolerant Networks in a Nutshell

Bachelor Thesis

Dave Wick
August 2007

Head:
Prof. Dr. Torsten Braun

RVS (Computer Networks and Distributed Systems)
Institute of Computer Science and Applied Mathematics (IAM)
University of Bern

Contents

1 Abstract

Within last decade mobility increased in almost every part of life. Current networking assumptions like low error-rates and short round-trip times may loose their entitlement and will have to be replaced by more adequate definitions. A *Delay Tolerant Network* (DTN) is a kind of Ad Hoc network with stronger and harder to fulfill restrictions. Not only random and unpredictable movement needs to be taken care of, as well facts like intermittent connectivity and possibly no available end-to-end path need a serious amount of attention. This tutorial names the various challenges of such a network and presents a framework developed by the DTNRG. The main contribution of this tutorial is the classification of several routing algorithms and approaches. The classification is based on how the routing is collecting the therefore needed information. We distinguish between *stateless, history-, location-, movement- and scheduling-based approaches* and present some algorithms for each category. To round up the overview we present and briefly explain two real-world examples.

2 Introduction

Within the last decade network communication has been in flux and many basic assumptions about how network links and applications behave required some adaptation. In network architectures where nodes are mobile and may suffer of outages and fast-changing connectivity, assumptions like short transmission delays, low data transfer error-rates or the existence of end-to-end connectivity no longer hold.

Ad Hoc networks try to cope with high node mobility and therefor frequent topology changes and as well with high error rates, energy and bandwidth-restrictions. Delay Tolerant Networks (DTN's) go further and make some even more constricting assumptions:

- Network partitioning

  Eventually no end-to-end connectivity between sender and destination is possible due to frequent or constant network partitioning. These partitions may occur because of geographical distance, lacking radio signal strength or other limiting factors.

- Network interruptions, failures and heterogeneity

  DTN's are often deployed in rough and adverse surroundings and nodes may be subject to numerous operation failures. These failures may cause network interruptions and disconnect beforehand linked nodes. Another assumption is that partitioned networks can be of heterogeneous structures, meaning not all local networks are using the same underlying protocols and applications.

- High error-rates

  Presumably short connectivity and high mobility in combination with weak signal strength and/or other aggravating circumstances is leading to a high link-error rate that makes end-to-end reliability difficult.

- Long & variable delay

  The intermittent connectivity causes long and hard to estimate delays. Data often has to be buffered or queued if there is no direct path to the destination node. Delays may last up to hours or days, depending on the mobility and connectivity within the network.

- Asymmetric data rates

  Due to the intermittent characteristics of a DTN most communication between two nodes will be mainly asymmetric. Elapsing time between request and answer will be, as mentioned before, rather hours than milliseconds.

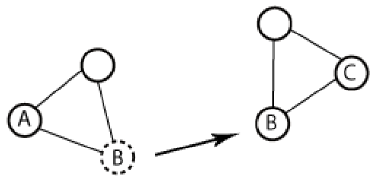

Figure 1: DTN data delivery occurs often through physical node movement. If node A wants to send a message to node C, the shortest path may lead across node B, which moves at a given time towards the range of node C.

- Energy, bandwidth, buffer & cost restrictions

  As within Ad-Hoc networks, energy is a limiting and valuable resource. Nodes may be turned off frequently to save energy and therefor weaken the already low connectivity. Other limiting factors are bandwidth bottlenecks and several cost restrictions regarding node-fitting. The available buffer-space restriction is far more limiting then in an ad-hoc network because most data can't be delivered immediately and directly to the destination and must be buffered at intermediate nodes.

The question arises under what circumstances these kind of networks might be found and/or be of any practical need. Although not too many real-world applications have been deployed so far, we can think of some interesting DTN-environments:

- Remote Area Networks

  Many remote villages in so called third- and second world countries suffer of internet and other communication connectivity. Current technologies like cable, wireless or satellite are often too expensive to install and operate or not available within these regions. The idea of using the available human or motorized mobility (i.e. regular bus services) to offer a kind of intermittent connectivity to the villagers seems to be an interesting approach. Read more about such networks in section 5.

- Military Battlefield Networks

  As most new technologies are somehow military-driven, DTN research is of interest to the army and emergency services which have to act and survive in difficult environments with usually low or no communication infrastructure. One might think of quickly deployed troops in remote areas with no centralized and stable communication. Soldiers can be equipped with low-range radios and exchange data within their group as well as messengers (i.e. soldiers, cyclists, parachutists) can physically carry data to other disconnected networks.

- Mobile Sensor Networks

  The almost legendary ZebraNet (see below) is a good example for a useful application of a DTN in combination with sensor nodes. Nodes equipped with sensors are often deployed in areas with no basic communication infrastructure. In some cases natural node-mobility can be used to collect and exchange sensor data and then be sent to a sink within range.

- Interplanetary Network

  It might sound like a chapter in the book 'A Hitchhikers Guide to Galaxy' [28], but there has been done some serious research about an interplanetary network. Planetary-routes can be calculated and encounters with other orbital objects are predictable. Data is transmitted if the planet or receiving object is within range or line of sight and data finds its way, hop by hop, through the galaxy. Not to mention that delivery delay needs to be measured in days rather than seconds.

Recently a few DTN-Projects have reached deployment status and have been tested and installed outside the laboratory. Some of them will be presented in section 5 (Real World DTN Applications).

This tutorial is organized as follows. Section 3 will introduce the concept of the DTNRG Framework which allows to mention DTN challenges and proposes an overlay structure to be used for further DTN protocol ideas and implementations. Section 4 classifies recently published or otherwise remarkable DTN routing protocols and tries to present an overview over the actual DTN research. Section 5 briefly presents two real world DTN applications; the ZebraNet which allows to track wildlife in their natural habitat and an the DakNet. Latter network makes rural villages in asia accessible to the internet. Section 6 concludes actual research status and tries to give an outlook to future progress and development.

3 The DTN Framework

As mentioned in the latter section a DTN has to cope with several challenges. These issues often correlate in a negative manner and have inherent *trade off* relationships. As in traditional and Ad Hoc networks objectives like high delivery ratio, low delivery delay and good scalability force researchers to optimize their protocols. Furthermore a DTN protocol often needs to be energy efficient, allowing nodes to switch off the radio for defined intervals to save battery power, use the available bandwidth and buffer with care and avoid chattiness of well-known internet protocols like TCP or DNS. Another objective which needs to be taken care of is that these protocols mostly have to work in cutoff regions with no possibility to update or repair nodes on-site. Therefor simplicity and robustness may play an important design role as well.

The Delay-Tolerant Networking Research Group (DTNRG) [22] proposes a framework that allows to design and implement new protocols. The framework makes some basic assumption about topology and requirements. These basic design decisions allow meeting the various and in the following section briefly mentioned DTN challenges.

3.1 DTN Challenges

There will be a necessity of DTN research for years, thanks to the following issues that need to be solved, optimized and combined in an efficient manner.

- Routing

  Routing is with no doubt the most challenging issue within a DTN. Many restrictions have to be taken into account and traditional network routing protocols will not satisfy the expectations. Links become available without any previous knowledge and the destination node may never be reachable immediately with a contemporaneous end-to-end path. Nodes change their routes randomly or may follow a predictable path, buffer and bandwidth restrictions force the protocol to send its discovery and topology information as sparingly as possible and avoid resource consuming flooding mechanisms.

  The routing algorithm further needs to know

  - when to send/forward a message,
  - where to send a message,
  - which message to send/forward,
  - which message to delete,
  - if it follows a single- or multi-copy strategy,
  - if it acts in a pro- or reactive manner,

  – the communication flow (i.e. node to node, node to sink etc.).

- Naming & Addressing

  Obviously the common Domain-Name System (DNS) can't be used within a DTN due to unacceptable long delivery delays. It may last hours or days for the DNS request to find its resolving destination and most likely the same duration to deliver the response back to the originator.
  Since the nodes are moving, a static naming system wouldn't be reasonable. Whereas the IP addressing scheme allows to follow a numeric path to locate the desired machine, a DTN node can be mobile and requires a more sophisticated locating mechanism. Long delivery delays make it even harder to find the destination node because currently it may have moved to another region. Even some kind of Home/Foreign-Location-Register (i.e GSM, Mobile-IP) will not be appropriate for most scenarios because of the additional requests.

- Data Transfer & Reliability

  As mentioned before, the internet is mainly based on the assumption of continuous, bidirectional, fast and reliable end-to-end paths with symmetric data rates. A protocol like TCP for example uses these characteristics to establish and release an end-to-end connection with a three- or even four-way handshake. To guarantee a reliable transport TCP segments are acknowledged on receipt by the destination. In a DTN which may not satisfy the assumptions mentioned above, such a 'chatty' protocol would seriously reduce the throughput and increment the delivery delay of the network. Simple, efficient and hop-by-hop reliable algorithms need to be found implying the intermittent character of such a network.

  Other hard to solve challenges are how to handle *fragmentation* and *flow-* and *rate control*. While pro- or reactive fragmentation may be accurate in sparse networks, the question arises, where to reassemble the fragments and what to do in case of loss of fragments. The issue of congestion control doesn't seem to be trivial as well: how should the network be informed about an acute congestion in such a delayed and resource restricted scenario? What can be done to route data along fast and reliable paths preemptively to avoid congestion at all?

- Security & QoS

  Security seems to be hard to implement in a DTN. Not only the authentication of the user and the integrity of the message is relevant, also forwarding permission and routers need to be checked and authorized. Network resources are limited so there's an actual need in restricting the usage of buffer and bandwidth. Furthermore, conventional cryptography methods like i.e. public-key signatures are inefficient due to long

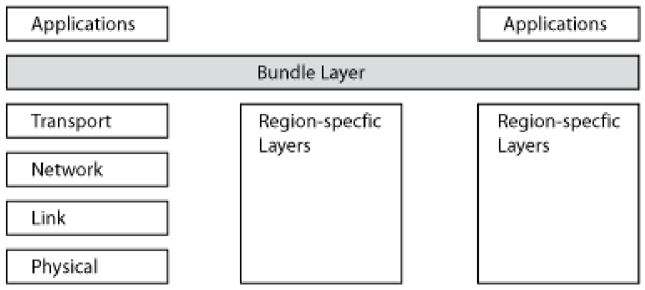

Figure 2: The DTN *bundle layer* is situated between the application and region-specific layers and allows communication across heterogeneous networks.

roundtrip times to fetch necessary certificates.
Quality of Service is another issue to be optimized in DTN's. Packets may be of different importance, one may think of alerts or software-updates for mobile sensor nodes which need to be delivered as reliable and fast as possible. Again questions of authentication arise because QoS might be bought and therefore be vulnerable to abuse.

- Time synchronization

  Many protocols require synchronized clocks for successful operation and in a field of energy shortage sleep and awake intervals of nodes needs to be well timed. While in wired networks a time-server may satisfy this necessity, a request and response to a central server would last too long and therefore be bare of any use what concerns time information.

3.2 The DTN Framework

The following section briefly presents an architecture for delay and disruption tolerant networks originally designed by the Delay-Tolerant Networking Research Group (DTNRG [22]). Recently this framework has been improved and extended and published as a informational RFC document [25].

The main part of the framework is the so called *bundle layer* which is situated below the application and above the transport layer of the network. DTN nodes use this layer to send and receive data bundles, optionally transfer reliable delivery responsibility from one hop to another or end-to-end acknowledgments. The naming scheme is kept flexible to allow interoperability and a basic security model is proposed to prevent unauthorized use of the network.

Bundle Concept

One of the main reasons to use a bundle concept, meaning that a complete data entity is bundled into one potentially large message, is to not split semantically related data into separate small-sized packets. In a DTN transfer delays may differ and in case of the loss of one packet the already transfered data can't be used either. As well from the point of view of management overhead, the transfer of one large data bundle is more efficient than the one with many small packets. Bundles may still be fragmented in case of small bandwidth or short contact duration between links and reassembled later at any node and anywhere in the net.

A bundle contains all necessary information required for routing and delivery such as the source and destination endpoint, an originator time-stamp, time-to-live, length and class-of-service value. Furthermore a 'report-to endpoint' may be specified for diagnostic and administrational reasons which allows to send report messages to another node then the source. Applications sending data over the network may as well request some kind of status or confirmation notifications i.e. *custody transfer requested, report when bundle delivered, report when bundle acknowledged by application* etc.

Because of the intermittent character of a DTN, there is in general no end-to-end connection between source and destination. Therefor bundles need to be forwarded and then stored at the next hop. This *Store And Forward* mechanism requires some persistent storage to survive long duration network outages and node restarts.

Nodes & Endpoints

A bundle source and destination is specified by an *Endpoint-Identifier* (EID). Such an EID may consist of one or more DTN nodes and the bundle is considered to be delivered successfully, if at least one (unicast, anycast) or all (multicast) nodes of an endpoint have received the bundle. An EID is expressed as an Uniform Resource Identifier (URI) and can be for example a DNS name or an *expression of interest* and is in general not related to routing or topological organization. In an earlier concept regions have been introduced; a full address consisted of a region id and a node specific entity id. The idea has been overtaken and generalized with the introduction of URI schemes. Both allow late binding meaning that URIs may be (re)interpreted from hop to hop. In terms of the regions concept a bundle first follows the route to its region and only when the bounds of the region have been reached, the second part, the entity id, is bound to the actual destination node.

Custody Transfer

The bundle layer provides a simple and unacknowledged delivery service, but the framework it self proposes two possibilities to get enhanced reliability: *end-to-end acknowledgment* (see above) and *custody transfer*. The latter method allows to transfer reliable delivery responsibility to a DTN node which is preferably closer to the destination node. Under certain circumstances this mechanism allows better usage of resources (i.e. the source can release it's memory used for eventual retransmission of the bundle) and even end-to-end transmission may speed up because of shorter and more reliable custody-to-custody node paths and therefore more successful bundle transmissions.

Classes of Service

To propose a basic QoS, three classes of delivery importance are introduced: *Bulk bundles* are transmitted with least effort and only if there are resources available. Whereas *normal class bundles* are prior to bulk bundles, but less important than bundles sent with the highest class of service, *the expedited bundles*. This service hierarchy only applies to bundles sent from the same endpoint, i.e. a bulk bundle may still be chosen from the queue even if there are higher priority messages in the same queue but originated at an other endpoint.

Routing

Routing in a DTN is an active topic of research and the framework only provides basic functionality and requirements to implement such an algorithm. The framework supports various kinds of delivery like *unicast*, *anycast* (deliver message to at least one node of a group) and *multicast* (which includes broadcasting as a special case).

Because contact time, duration and volume between two or more nodes are not necessarily known precisely ahead of time, routing is especially challenging in a DTN. To support the calculation of the route the framework distinguishes between multiple types of contact:

- *Persistent Contacts*. These contacts are always on and available.
- *On-Demand Contacts*. On-demand contacts are potentially available but require some action to initiate the contact. One may think of a dial-up modem where the connection is set up if required and then persistent until released.
- *Intermittent Contacts*. In contrast to on-demand contacts, intermittent contacts are dependent from other, non controllable reasons. Intermittent contacts can be *scheduled, predicted* or *opportunistic*. Whereas *scheduled contacts* follow a certain rule and time and duration of contact can be calculated in advance (i.e. in-sight time of low-earth orbiting satellites), *predictable contacts* may be estimated or, oh, predicted for example based on previous encounters or movement patterns, but not calculated exactly. *Opportunistic contacts* are of random and unexpected occurrence and duration. One may think of two pedestrians equipped with DTN

nodes crossing their ways and therefor coming within radio range for an undetermined amount of time.

Security

The framework proposes an optional security architecture called DTNSEC that uses hop-by-hop as well as end-to-end authentication and integrity mechanisms. The main issue is to be able to handle access control to the network and forwarding control of the messages. No unauthorized nodes should be able to send data over the DTN and no data should be forwarded at nodes not part of the network.

4 Routing & Information Diffusion Approaches

This chapter discusses approaches to solve the DTN routing & information diffusion challenges. Many solutions have been proposed and a wide range of possible approaches has been brought onto paper. Routing is mainly a question of how to forward a message to the destination. But as simple as it may sound, the challenge is enormous and includes different closely related issues as mentioned in section 3.1 (DTN Challenges).

In general, routing has to find a good path to a designated endpoint but concurrently to deal with a resource shortage. In an optimal case all data can be delivered and the protocol finds the fastest and shortest path between the two involved nodes. In the real world or even in simulations restrictions occur or are defined and therefor initiate the need for economical usage of resources. Depending on the application using the DTN, it can be useful to drop packets and free buffers quite early to give newly sent packets a good chance to be delivered in time while, on the other hand, it may be important to deliver as many packets as possible, no matter how long it lasts.

While reading the papers published on behalf of DTN-Routing, one will recognize the variety of approaches and different weighting of objectives. Even though all in the following chapter presented ideas can be called kind of 'routing protocol', they often use different models, assumptions and communication flows and therefore prevent a detailed comparison. Let's just have a quick look at the broad variety of mobility models; simple random walk models, where nodes define a random target point and their walking speed and then try to reach this anchor point straightforward. Other models use predefined or somehow structured paths and limit speed and resting time to certain bounds. More sophisticated models try to adapt human behavior and let the nodes circulate in so called *social orbits* [15] or according to some pre-evaluated movement patterns.

Although most approaches use some kind of probability function or metric to decide whether to forward or not the data-packet, they highly differ in which values and properties have been taken into account. We decided to classify algorithms based on their source of routing knowledge. We can distinguish between five types of approaches. Many algorithms combine two or more approaches but focus on one specific topic to optimize.

4.1 Stateless Algorithms

Most propositions are based on a stateless routing algorithm, meaning in general no further (past or future) location or contact data is needed to make the forwarding decisions. Fundamental research on replication & erasure coding has been done with stateless algorithms and can be used and combined with more complex and involving strategies.

The following presented approaches are purely stateless while most algorithms in the subsequent sections extend these mechanisms by using more indicators.

Epidemic Routing

No doubt, Epidemic Routing [19] is the most simple routing approach and can be called a broadcasting algorithm. A node spreads its messages through the net by exchanging a *summary vector*, a hash-table that contains a signature of all locally available messages, with another node within range. Comparison of the received with its own vector determines which messages to exchange. To avoid redundant connections, nodes cache the time and node-id of the last meeting and do not reconnect for a given time period. A message contains three additional header fields, namely a *message id*, a *hop-count* and an optional *acknowledgment request* field. The hop-count field is the only transmission limiting field and can be compared to the TTL field in IP packets. In contrast to the TTL, the hop-count field indicates wether to send a message to a neighbor node or not. A value greater then one forwards the message to any interested node while a value of one indicates, that it might be send to no other node than the destination. The request of acknowledgment can be set and causes the destination node to send or piggyback an acknowledgment message upon reception.

Delivery rate and delay depend straightforward on available buffer space at each node, the range of the radio and the chosen *hop count* value. A simple first-in-first-out (FIFO) buffer management is proposed and works well as long as the buffer space exceeds the number of messages to deliver. Otherwise more sophisticated algorithms are needed to be implemented. While this broadcasting algorithm allows to find the shortest route to the destination node, a huge resource consuming overhead has to taken into account. Epidemic Routing is often used as a benchmark for delivery delay for more complex routing approaches.

Using Redundancy to Cope with Failures in a DTN

This approach [2] focuses on the issue how to split, replicate and use erasure codes to achieve an optimal message delivery probability over multiple paths. Failure probability of every available path is assumed to be known and can be *bernoulli-distributed*, meaning either all or none of the data of a block can be transmitted over a certain link, or *gaussian*, meaning partial data may be transmitted and used as well (i.e. in case of an unforeseen link interruption).

Bundles are split into blocks according the formula $b = \frac{(mr)}{l}$, where b is the number of blocks, m the size of the bundle, r the replication factor and l the size of one block. If $\frac{b}{r}$ or more blocks are delivered to the destination, the bundle can be decoded successfully thanks to the erasure code. The question what fraction should be sent on which path is complex and in certain cases even NP-hard. The authors use the *mixed integer program* (MIP) to calculate the optimal division, but already with a small number of paths computational complexity is unacceptably high. A good approximation has been borrowed from the modern portfolio theory. The *Maximum Sharpe-Ratio* algorithm is, once derived, easy to use and accounts even hard to calculate path dependencies and volume constraints. In few words, the *Sharpe-Ratio* approximates a so called *efficient frontier* [26] that represents the line of, in terms of asset management, 'highest return for a given risk', which can easily be adopted to questions of replication and path selection.

It's been shown, that the *Sharpe-Ratio* outperforms other, more simple approaches as for example *simple replication*, which duplicates the whole bundle and uses no erasure coding, or *proportional division*, meaning encoded blocks are allocated for a path proportional to its delivery probability. Especially in environments with forced fragmentation the *Sharpe-Ratio* algorithm proves its strength and robustness.

Network Coding for Efficient Communication

Network Coding [4] is a relatively recent field in information theory. This technique allows combination of multiple symbols into one data-set. In terms of DTN, a bundle may be sent out from a node as a linear combination of previously received bundles. The main advantage results in reduction of the retransmission overhead in comparison to other probabilistic routing algorithms.

If a node receives a packet, it includes the data into the matrix, presumed it increases the rank of the set of received packets at this node. If a packet does not, it has been seen by this node already and will be ignored. A combined information vector is broadcasted on reception of such an *innovative* packet to the neighbors with probability *d*. This parameter is borrowed from probabilistic routing and can be used to control flooding and replication of data. To limit the size of the matrix, data is grouped into *generations*. A hashing function ensures a maximum matrix size and allows adjunct assignment of data to a generation. Once a node has decoded the interesting data it is of no further use to this node, but neighbors still might be interested in that piece of information. The concept of *information aging* allows to stepwise reduce the rank of the matrix after a successful delivery and therefore reduces the probability of further broadcasting.

Simulations have shown that network coding allows reliable and fast dissemination to all nodes with a significantly lower resource overhead compared to other probabilistic approaches. Coding and decoding is feasible even to very limited memory and processing power.

A Hybrid Routing Approach

This paper [7] proposes a combination of replication and erasure coding approaches. While replication speeds up propagation of bundles through the network, erasure coding tends to be more robust in environments with poor and unstable connectivity.

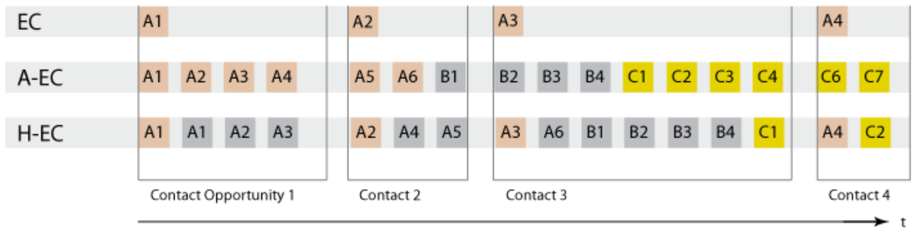
EC
A-EC
H-EC
A1
A2
A3
A4
A1 A2 A3 A4 A5 A6 B1 B2 B3 B4 C1 C2 C3 C4 C6 C7
A1 A1 A2 A3 A2 A4 A5 A3 A6 B1 B2 B3 B4 C1 A4 C2
Contact Opportunity 1
Contact 2
Contact 3
Contact 4
t

Figure 3: Different message dissemination approaches based on*replication* and *erasure coding*.

Multiple methods of bandwidth allocation are examined. With simple *erasure coding* (EC) the number of transmitted blocks is limited to one for each contact, regardless of the length of the connection between two nodes. This approach obviously works well, if the contact duration is not much longer then the time required to send the data blocks. Otherwise, if most network contact time is longer, available connectivity isn't used optimally and EC results in inefficiency. An *aggressive forwarding* feature aims this problem by continuously filling up the available time slots with blocks. As seen in figure 3, the so called A-EC scheme doesn't waste any contact duration, but tends to poor delivery ratio and/or large delivery delay when most employed nodes are either unreliable or hardly move closer to the destination. The *hybrid* scheme (H-EC) effectively combines the advantages of replication *and* erasure coding. With a replication factor $r = 2$, one might think of two stacks, each containing a copy of all messages (i.e. A, B, C) to send, respectively their blocks (A1, A2, etc.). While the first slot in an upcoming contact is used to send a block from stack one, the remaining available slots are filled up with blocks from stack two. This approach better utilizes the contact opportunity and still remains robust in case a contact is not able to deliver its blocks to the destination because of the guaranteed block distribution to multiple forwarding nodes. Since there are only two copies of each message sent, the traffic-overhead is limited to twice the amount of the simple EC scheme.

As expected, the H-EC scheme performs almost identically as A-EC in networks with high connectivity and low average delivery delay, with only a moderate overhead. Even in networks with frequent interruptions, low node density and possible node failure, the H-EC approach still performs very well, although not as good as a EC scheme with a higher replication factor r then two. A proposed improvement could be (not yet specified) *adaptive coding* which constantly checks the network status. I.e. in case of poor connectivity r might be increased to achieve the same level of reliability. On the other hand one should not forget the costs and overhead of the realtime status-measurement.

Opportunistic Content Distribution in an Urban Setting [8]

This paper examines how intermittently and partially connected mobile nodes can accelerate and improve information diffusion in a target group. Source of dissemination are fixed nodes, while mobile nodes are targeted and may act as data relays. To probe different approaches on realistic data, mobility traces of several students have been gathered in a real world experiment. 20 stationary devices, spread all over the city of Cambridge UK, covering places like student-labs, grocery shops or restaurants, and 40 mobile devices have been logging bypassing devices. The collected data then has been grouped into three kinds of connections: *inter-mobile*, *mobile to fixed nodes* and *external* contacts. While contacts between mobile nodes often occurred during weekdays due to the experiment setup, the mobile to fixed nodes contact frequency wasn't as high as expected. This may be because of inappropriate deployment of the fixed antennas or because of gaps between beaconing intervals. Interestingly enough, data logs of bypassing external devices have been merged and the authors revealed that many external contacts connected as well to one or more other nodes participating the experiment and therefore played an important role in performance evaluation of the following information diffusion approaches.

Goal of this routing approach is to broadcast messages from fixed source nodes to as many as possible mobile devices of a specified target group. Multiple flooding strategies have been considered being efficient, such as:

- *Selfish*, where nodes get their data from the fixed access point and never pass it on to other nodes.
- *Collectivist*, where nodes transmit messages to other nodes within the same target group.
- *Extended collaboration*, same as *collectivist* but furthermore external contact may act as relays and deliver data to mobile nodes. External contacts are chosen either dependent of their number of contacts or bridging connectivity between adjacent pairs of fixed/mobile stations.
- *Top n students*, where only the top n nodes, in regard of the number of contacts, are allowed to forward messages.
- *Strangers only*, where only external contacts can relay the message.

Obviously the *selfish* and *strangers only* strategy performs poorly compared to the other schemes. Best results have been achieved with extended collaboration, especially in networks with a small number of fixed access points or mobiles nodes within a target group.

Spray and Wait

This replication based algorithm [12] consists of two phases: first, a number of copies is *sprayed* into the network and then it *waits* until one of these nodes reaches the destination. There are different spraying methods, two of them are mentioned and examined: one is the simple *Source Spray and Wait* method, where a node forwards L copies of a message to the first L distinct nodes it encounters. A more sophisticated approach is the so called *Binary Spray and Wait* method, where always half of the available copies at a certain node are forwarded to a neighbor node (with no such copies) until only one message is left at each node. If all messages are spread to L distinct nodes, no more forwarding occurs and only direct transmission to the destination node is allowed.

The algorithm guarantees controlled replication and one of its advantages is, that even in large networks the replication remains constant unlike in other, probability based approaches. The main issue is to calculate the number of messages L required to satisfy a certain level of maximum expected delay. This delay is expressed as a factor a times the optimal delay which can be reached through epidemic flooding. Interestingly, L is neither dependent on the network size nor on the transmission range of the devices, but solely depends on a and the number of nodes.

Under low traffic *Spray and Wait* performs nearly as good as *Epidemic Routing*, but with an order of magnitude less transmissions. As the traffic load increases, it outperforms all compared protocols in terms of delay and transmission overhead. Even in different connectivity scenarios, the approach seems to be robust and efficient. While simple flooding mechanisms work well in low connectivity environments, the network overhead reaches unacceptable high values in high connectivity networks. *Spray and Wait* benefits of it's limited contention and efficient message dissemination.

4.2 History-based Algorithms

These algorithms use data of the past to find an efficient route to the destination node. Logged data can be a history of *recent encounters* with other contacts as well as for example *contact-time*, *contact-frequency*, *contact-time-location* tuples. History-based algorithms usually need some time to *warm up*, but in return often are more adaptive to topology and mobility changes.

Practical Routing in DTN's

The main idea of this approach [1] is to use of a metric called *minimum estimated expected delay MEED* to determine an short route to the destination. It calculates the estimated waiting time to encounter a certain node based on the observed contact history. Previous work has shown, that such estimations can make predictions with accuracy greater than 80%. A sliding window mechanism records the previous encounters and changing the windows size allows to adjust the 'adaption to topology changes' speed.

These *MEED* values are used to feed the local link state table which then is broadcasted with epidemic flooding through the network. Not many DTN algorithms use *Link-State Routing* due to its high expected administrative network overhead, but with some slight modifications it may be beneficial to use such a protocol. To not overburden the network, the link state update packets may be suppressed in case of minor changes. If an update packet needs to be sent, the flooding algorithm is easily initiated and very robust, declared at the price of some more network traffic. An other advantage is that in case a new node joins, it is able to get the whole topology information from one single neighboring node. As a further modification, the route of a data packet is not determined at source (*source-routing*), instead only the next hop will be calculated. After successful delivery to this hop, the current node may recalculate the topology based on the local available information and find a more accurate route. This *per-hop-routing* mechanism and the fact that DTN's may be split up in multiple disconnected regions can lead to loops under certain circumstances, because the local topology information may differ.

The usage of this MEED based forwarding criteria lead to a delivery rate of more then 95% in relation to simple flooding, but with only one single copy per message, and performed nearly as good as protocols with complete knowledge of the topology.

DFT-MSN

This multi-copy scheme [5] is another kind of probabilistic routing and consists of two key components: *data transmission* and *queue management*. Messages are assumed to find its way through the network, which consists of N uniformly distributed nodes, to one of the n sinks. To get an optimal trade-off between data delivery rate/ delay and transmission/buffer overhead, the routing is based on two parameters:

- The *Nodal Delivery Probability* indicates the likelihood, that a node can deliver messages to the sink. A probability value p is initialized with 0 and incremented in case of successful message delivery. If there's no message transmission within a certain interval, the value is decremented.
- The *Message Fault Tolerance* represents the amount of redundancy of a message in the network. To define this value, two approaches are possible. First, a simple *hop-count* based approach may be used, where the hop-count indicates the number of times a message has been copied and forwarded and therefor is a good guess for its redundancy. Second, a more complex, but as well more accurate approach is to calculate the *message fault tolerance* based on the delivery probability. After initialization with 0, the fault tolerance is incremented according to the delivery probability of the supplied neighbors.

The *queueing component* takes care of the available buffer and sorts the messages ascending to their *message fault tolerance* value. This guarantees that 'fresh' messages are favored and will be transmitted prior to older messages. If a messages redundancy value exceeds a configurable threshold, the message

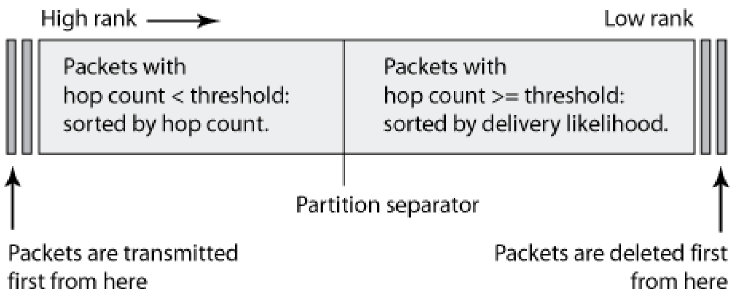
High rank
Low rank
Packets with hop count < threshold: sorted by hop count.
Packets with hop count >= threshold: sorted by delivery likelihood.
Partition separator
Packets are transmitted first from here
Packets are deleted first from here

Figure 4: The MaxProp queue management strategy.

will be dropped and removed from the queue even the buffer is not full. In case no more memory is available and all messages fault tolerances are below this threshold, the message with the highest value will be dropped. This queue management scheme allows to determine the available buffer space based on the fault tolerance threshold.

Buffer space information is important for the *data transmission component*. Nodes moving into transmission ranges of other devices, exchange their delivery probabilities and available buffer space via hand-shaking messages. The decisive algorithm now determines a set of neighbors based on their space, threshold and probability information and forwards the bundle to them.

Beside the required warm up phase to adjust the contact probabilities, this algorithm seems to be quite robust and independent of a specific mobility model. Simulation results show that this scheme achieves good performance in order of delay and throughput with an acceptable transmission overhead.

MaxProp

The *MaxProp* [11] approach is somehow similar to the latter presented *DFT-MSN* scheme. The queue management consists of a combination of the formerly mentioned methods to calculate the so called *Message Fault Tolerance*; namely the *hop count* and the *delivery likelihood* based approach. While the first partition of the buffer is sorted according to the ascending hop count value, the second partition is filled with messages exceeding a certain hop count threshold and sorted in order of their delivery probability. Assumed the network includes s nodes, the likelihood for node i to meet one of the other nodes is initialized to $\frac{1}{(s-1)}$. When node j is encountered, the probability value to meet node j is incremented by 1 and then all values are normalized. This *incremental averaging* gives lower values to nodes infrequently seen et vice versa.

While this ordering has the advantage that 'fresh' messages are given a *head-start* because of their low hop count, other mechanisms to improve performance like *systemwide acknowledgments of delivery* and *routing information exchange* between neighbors are involved. An interesting tweaking parameter is the logical partition of the available buffer. The authors propose to prioritize low hop count packets in case where the average number of per-contact transferred bytes x is much smaller than the available buffer space. As x grows, the threshold should be reduced to give more space to delivery likelihood ordering.

MaxProp has been tested against other approaches, namely a *Oracle-Based Dijkstra* algorithm, where the shortest path is calculated based on future and globally known encounters, and a *Most Encountered/Drop Least Encountered* approach, which consists only of the sorting mechanism based on delivery probabilities described above. In the simulation setup with different network traffic scenarios and buffer sizes, *MaxProp* outperformed both algorithms what concerns delay and throughput, surprisingly even the oracle-based one, although it has the knowledge of all future contact encounters.

Enabling Disconnected Transitive Communication (DTC) in Mobile Ad Hoc Networks

This paper proposing a *DTC* [17] algorithm has been written back in the year 2001, but still offers some interesting thoughts and approaches. No implementation and simulation of *DTC* ever took place and therefore it must be seen as a purely theoretical approach.

The idea is well known; it is to forward messages to the one neighbor, which is as 'close' as possible to the destination. The distance metric may be based on five components and can be calculated with a configurable *utility function*:

- *Most recently noticed*, which is a FIFO queue containing node information about recent encounters. The *TimeOut* value is proposed to normalize results between hosts.

$$U_{MRN} = (1 - \frac{CurrentTime - TimeLastNoticed}{TimeOut}) * 100$$

- *Most frequently noticed*, which extends the U_{MNR} by a counter to record encounters with a certain node.

$$U_{MFN} = (1 - \frac{CurrentTime - TimeLastNoticed}{NumTimesNoticed * TimeOut}) * 100$$

- *Future plans*, which, in contrast to the previous and following calculations, require some user interactions to reveal its information. One may think of interfaces to the devices calendar-applications or a simple list of hosts which are expected to meet within a given time frame.

$$U_{CAL} = (1 - \frac{NextMeetingTime - CurrentTime}{TimeOut}) * 100$$

- The available *Power* or battery level can be used to supply valuable information to the utility function.

$$U_{Power} = (1 - \frac{TimeOut}{EstimatedRemainingPower}) * 100$$

- *Rediscovery interval*, which defines the time interval to send beacons to its neighboring nodes. This value may vary during lifetime and be an indicator for the state of the network topology. In case of rapidly changing neighbors the RDI is increased, while it decreases in low connectivity and slowly changing networks.

$$U_{RDI} = \frac{Min_{RDI}}{RDI} * 100$$

The *utility* may be calculated in two ways. First, a node probes its neighbors and then collects the previous mentioned data of every node to calculate the utility locally. Second, a node distributes its source defined utility function and lets each interested node be in charge of calculating the utility. The second approach takes into account, that data may be private and the computing load is distributed over multiple devices. Furthermore less data is sent through the net and therefor the overhead can be minimized. On the other hand, collection of data may be useful if it is cached and reused for future utility calculations.

4.3 Location-based Algorithms

Nowadays it is no longer a challenge to locate the actual position of a node. GPS devices are cheap and small and can be used to frequently update the location information. An other way to determine its location is to use fixed infrastructure like antennas to trace the device mobility based on signal strength, although some questions about privacy might arise. The following theoretical analysis of how location data can be used to determine an optimal route for a message includes in all cases further stateless or history-based approaches. Purely location based algorithms proved not to be very effective [16].

Sociological Orbits for Efficient Routing

The authors of this paper [15] present a probabilistic mobility model where the nodes move between a set of *hubs* within different *sociological orbits*. Such mobility profiles have been extracted from the observation, that the movement of a user includes repetitive patterns involving a small number of places, so called *hubs*. As an example, the university campus would be an *orbit*, while spots like the lecture room, research lab or cafeteria are several hubs within this orbit. Users now move within and between these hubs and according to their mobility profile, one can generate a *transition probability matrix* for each user.

This transition matrix and two other parameters are used for the following routing algorithms: the *contact probability* defines the chance of two nodes ever get within direct transmission range and the *delivery probability* describes the chance of successful message delivery from the source to the destination using all possible paths of intermediary nodes, that come in range. The paper proves, that these parameters can be calculated using a approximation function, however, computing the entire directed probability graph is of NP complexity. The proposed *Sociological Orbit aware Location Approximation and Routing* protocols (SOLAR) are:

- *Static SOLAR KSP* assumes every node to know the *transition matrices* of all other nodes. With Dijkstra's Shortest Path a *delivery subgraph* is computed and k shortest paths (KSP) are chosen, such that a path with minimum total weight is in favor and each path has a different next hop node from source. Ordered by their delivery probability, multiple strategies exist to forward the bundles on the k paths: Either a bundle is *sent to all*, only *sent to the best* or *sent to any* of the next hop nodes on the k best paths.

- *Dynamic SOLAR KSP*. In this variation, each node computes the KSP *locally* and sends the bundle to the k nodes from amongst its current neighbors with the highest delivery probability, which is in contrast to static SOLAR, where bundles are only forwarded to *next node hops* on the k paths. To reduce overhead, a node suppresses forwarding within the hub it received the bundle.

- *SOLAR HUB* forces the source node to send the bundle to its neighbors that have a higher *delivery probability to the hubs* visited by the destination node and not the node itself. Different forwarding strategies exist, i.e. a copy of the bundle can be sent to $\frac{k}{2}$ neighbors with higher probability to visit the 'most visited' hub of the destination, whereas the rest is sent to neighbors visiting the 'second most visited' hub.

Simulation results show that the SOLAR protocols clearly outperform epidemic routing, surprisingly also in terms of throughput and delivery delay. This may be caused due to the simulation setup (buffer size, message load) and as well, routing efficiency is bought by additional computing to calculate the *delivery subgraph*, but still proves SOLAR to be an economical and fast algorithm. Dynamic SOLAR and SOLAR HUB usually outperform the static version, which can be explained with the static path allocation and therefore missing ability to adapt to mobility and topology changes.

MV Routing and Capacity Building

MV routing [9] learns the structure in the movement patterns of nodes, based on *recent encounters* and *visits to locations.* After exchanging a list of carried bundles, two neighboring nodes forward data according to the *delivery likelihood.* This value represents the sum of all probabilities that a path to the destination can be found, meaning that involved hops visit the same region simultaneously at a given time. The probability calculation doesn't scale well, but fortunately a good approximation exists for larger topologies.

This rather well-known approach is extended by the use of *autonomous agents.* The problem of using agents to transport data is how to dispatch such a device to service requests optimally. Since this problem is reducible to the *dial-a-ride* [27] issue, it is proven to be NP-hard as well. The authors propose the notion of *null-space* to optimize multiple metrics, such as *bandwidth*, *message latency* or *peer latency*, which is defined as the average time since a peer was last visited by an agent. By creating a controller $Cont_x$ for each metric and the ordering according to their importance, an adequate hierarchy can be built. The *null-space* of a particular controller $Cont_i$ is defined as the set of actions that can be taken, without affecting the performance of $Cont_i$. MV routing now defines thresholds as a performance measure for the controllers, which permits null-spaces of sufficient size to optimize multiple objectives.

While the simulation assumes a network consisting of mobile users and geographically fixed destination nodes, the implementation of agent-augmented routing improved delivery rate and latency metrics by up to 50%, but needs to be related to the specific simulation setup and the small number of nodes (14) and agents (2) within the network.

Routing in a Mobility Pattern Space

This paper [10] defines a generic DTN routing scheme using a multi-dimensional Euclidian space that represents the mobility pattern of a node. This *MobySpace* helps nodes to take routing decisions. In general, it is a good idea to forward bundles to neighboring nodes with a similar mobility pattern like the destination node, because the chance increases, that this node will meet the actual destination. The coordinates of a pattern within the *MobySpace* define a *MobyPoint*, which is used to move the bundles towards nodes with their MobyPoint closer and closer to the MobyPoint of the destination.

One might think of many methods to define the dimensions of such a MobySpace. The authors present a simple but useful space that uses a fixed or at least limited number of axis. Nodes visits to particular locations should be tracked and every location is defined as one dimension. Frequency and duration define the coordinate on the axis and represent the probability to find the node at this location. Recent studies show that users typically visit only a few access points frequently and for long periods while visiting others rarely and quickly. This characteristic promises distinct MobyPoints and allows to define a mobility model for the simulations. A possible limitation of such a routing is, that a bundle may reach a local maximum, meaning getting to the most similar Mobypoint in the local neighborhood, but cannot reach for any reason the actual destination node.

Multiple metrics to measure the distance between two MobyPoints are proposed:

- *Euclidian distance*, which is the most common metric, defined as the root of the sum of the square differences between the coordinates of a point.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

- *Canberra distance*, which is the sum of a series of fractional differences between coordinates of a pair of points.

$$d_{ij} = \sum_{k=1}^{n} \frac{\mid x_{ik} - x_{jk} \mid}{\mid x_{ik} \mid + \mid x_{jk} \mid}$$

- *Cosine angle separation*, which represents the cosine angle between two vectors.

$$d_{ij} = \frac{\sum_{k=1}^{n}(x_{ik} \cdot x_{jk})}{\sqrt{\sum_{k=1}^{n} x_{ik}^2 \cdot \sum_{r=1}^{n} x_{jr}^2}}$$

- *Matching distance*, which is defined as the raw number of location probabilities that are similar (i.e. the absolute difference is smaller than δ).

Simulation results have been compared to epidemic, direct and random transmission algorithms. While the Euclidian distance and cosine angle metric lead to very good results, canberra and especially matching performed just a little better then direct and random transmission. Clearly, epidemic routing outperformed all approaches in order of delivery delay, although it is the only multi-copy scheme within this comparison. Interestingly the performance results didn't change significantly in different scenarios of knowledge about the destinations MobyPoint.

Locating Nodes with EASE / Last Encounter Routing

This pioneering paper [14] shows how nodal mobility can be used to disseminate destination location information without adding any communication overhead. Although it has been written in the context of mobile ad hoc networks, it is useful for DTN routing as well. The idea behind this location service is that each node maintains a local database of the *time* and *location* of its last encounter with every other node. No more additional or auxiliary information is needed for a message to find its destination node within the network with an acceptable number of hops. Amazingly, the length of the routes with this *Exponential Age SEarch* EASE algorithm are of same order as the distance between source and destination, meaning length is within a *constant* factor of routes with perfect location information.

The EASE algorithm does not prescribe any routing mechanism to reach the destination, it solely describes a way to locate the destination. *Last encounter routing* (LER) profits of and in same time requires three network characteristics to work efficiently: *locality* is necessary to ensure a node can be found somewhere close to a location it has been seen recently. There is no point in using LER if a node can jump to the other end of the topology within seconds. *Mixing* of node trajectories is required to ensure diffusion of information about the destination node within its neighborhood. And last but not least, *homogeneity* in the mobility process is necessary to guarantee that location information spreads at least as fast as the destination node moves. The algorithm itself is explained quickly: If a message needs to be sent to a destination with unknown location, a search for the destination's last encounter is initialized within the sources' neighborhood. The source then uses the 'freshest' i.e. most recent encounter-record and forwards the message to this position. The same procedure is repeated when the packet reaches the new anchor point. Because of the decreasing age of the encounter-records, location estimation is becoming more precise and the destination node may be within reach after a few hops. An extension, the greedy EASE algorithm (GREASE), checks the the age of the last encounter at every hop it passes, not just as it reaches the newly defined anchor point. This approach is more flexible, and in case it follows a 'hot' trajectory, even less costly due to no more search phases are required.

Simulations have been run to evaluate the quality of the EASE/GREASE routes compared to full knowledge and based on different mobility models, but the variety and complexity of the tests exceed the scope of this tutorial. An interesting plan for future implementations is to not only use nodal encounter data, but also force packets to trace their route and encounters to update aged information at bypassed hops.

4.4 Movement-based Algorithm

Some routing ideas are based on feeding the probability function to determines forwarding decisions with movement and velocity data. In simulations purely movement-based algorithms perform quite poor, but can add valuable information in combination with other approaches.

Knowledge-Based Opportunistic Forwarding MoVe

The MoVe [16] algorithm uses velocity and direction information to make opportunistic forwarding decisions. All nodes are assumed to know their own position and velocity. Furthermore the location of the destination is immovable and globally known. The authors of the paper propose three simple algorithms:

- *Location-Based*, which is a form of greedy and geographical-based routing. Messages are only forwarded, if the neighboring node is closer to the destination then its own position. See the previous section for more *location-based* algorithms.
- *MoVe*, which calculates the estimated closest distance *cd* to the destination, if both nodes continue on their route at the currently detected direction. Note, that in case of both nodes approaching the destination, only *cd* is taken into account, but in case of moving away, velocity decides which node has to carry the message; obviously the device that moves slower away from its destination.
- *MoVe-Lookahead* extends *MoVe* by adding the nodes next way-points, where their trajectories change, to be known. This future information

allows to calculate a more appropriate *cd* in case the nodes moves away from its predicted route before reaching the pre-calculated *cd*.

In the simulation setup where nodes move in piece-wise linear fashion, following typical city street structures, both *MoVe* algorithms outperform the location-based approach, but clearly fall behind flooding what concerns delay and throughput. On the other side buffer and transmission overhead is moderate because of their single-copy strategy. Interestingly the *lookahead* approach doesn't significantly improve the overall performance of the system. The authors admit, that the results strongly depend on the setup and mobility models. An other simulation with real-world bus traces lead to a result, where the purely *location-based* algorithm outperformed both *MoVe* approaches.

4.5 Scheduling-based Algorithm

Approaches based on scheduling are mainly of supportive nature and so far no routing algorithm is purely based on scheduling principles. The following paper therefore simply examines different scheduling strategies of messengers to support data transfers between disconnected networks.

Interregional Messenger Scheduling

This paper [3] proposes the usage of dedicated messengers to relay messages between disconnected regions. These regions can be stationary or mobile, as well as the nodes within a region. 'Traditional' DTN forwarding may still occur, but the goal is to understand how messengers can additionally improve network performance within such disconnected networks. Two typical DTN assumptions, the so called *node blindness* and *node autonomy*, no longer hold in order to allow the efficient usage of messengers. This approach requires the region, or at least one node belonging to the region, to know its location. Furthermore the source nodes need to know within which region to find the destination node. These informations might be available through GPS and some low capacity long-range communication technology and are assumed to be always available.

A messenger may belong to a region or be independent. *Regional Messengers* are owned by a region and carry data from this region to an other and then immediately return to its origin, possibly carrying some data as well on the way back. In contrast *Independent Messengers* are not bound to a certain region and are managed by the region it currently resides. Once delivered the data at the destination region, the messengers can be instructed to carry bundles to an other region. While this independency allows to react more flexible to changing data flows, it may also suffer from starvation in case a region needs to send data, but no messenger is assigned to deliver bundles to this region.

Three different scheduling strategies are proposed: *Periodic Scheduling* charges messengers to leave to certain regions at pre-determined times, even if there is currently no data to transfer, while *On-Demand Scheduled* messengers leave as soon as there is any message to be delivered to an other region. *Storage-Based Scheduling* makes the messenger to move towards its destination as soon as it has a predefined storage capacity filled with data.

The authors admit, that the efficiency of messengers is strongly dependent on the environment they are deployed into. As the *On-Demand* strategy promises a minimal delay under low or moderate message generation rates, it is expensive in terms of the number of trips the messengers have to take. *Periodic Scheduling* seems to be a the happy medium, as it provides a close to on-demand delay with reasonable costs, but it might be tricky to define the optimal time period. *Storage-Based Scheduling* provides the highest efficiency and least cost but comes at the expenses of very high delay, especially at low network traffic.

4.6 Further Classification Criteria

One might choose different classification criteria, depending on what is intended to be shown. For the aim of presenting a general overview of actual DTN approaches, the chosen criteria seems to be sufficient. Other criteria might be needed to reveal different comparison results.

Approaches could be distinguished as well in respect to

- their replication & erasure coding behavior,
- the type of contacts used to predict the routes,
- their topology knowledge,
- their underlying mobility models,
- their objectives.

5 Real World DTN Applications

This chapter presents two real world applications of a delay tolerant network. Although many ideas exist how and where to use DTN protocols, so far just a small number of projects has been deployed. Two of them, the *DakNet* and the almost legendary *Zebranet* will be briefly explained in this chapter.

5.1 Low-Cost Communication for Rural Internet Kiosks Using Mechanical Backhauls

The application [6] allows to set up simple and cost efficient connectivity for rural areas. Based on a *kiosk-system*, the formerly deployed DakNet [23] has been optimized and extended to cope with nodal mobility. Although this system hasn't been deployed so far it is planned to use it at large scale in india.

While *rural kiosks* may be of great benefit for people living in small and remote villages, providing them with basic services as birth, marriage and death certificates, land records and other information, it is often difficult and very expensive to set up the therefore required connection to the internet. Either the connection is slow and unreliable (dial-up line) or it is expensive and needs an extensive setup (satellite terminal). This paper proposes the usage of buses and cars, so called *mechanical backhauls*, to transfer data between kiosks, mobile users and internet gateways in larger cities. The architectural design includes naming, addressing, forwarding and routing and has the prior goal to allow cheap, reliable, mobile, secure and interoperable communication in such a disconnected network scenario. Only a few and widely available technologies are required for deployment: *cheap storage*, which is used at kiosks and backhauls, *wireless network cards* to interconnect the nodes at a given opportunity and a *cellular overlay*, i.e. an ubiquitous GPRS-like data service at low bit rates, to implement a separate control layer. While application data is transferred using opportunistic links to the backhauls, the control plane provides routing updates to optimize the use of the data plane.

Naming & Addressing

Every user within the network gets a globally unique id (GUID) which is a hash of his telephone number or email address and registers at least at one DTN Router as his *custodian.* A Custodian has multiple functions, it stores bundles on behalf of a user to be picked up later or acts as an anchor for mobile users. The full address is represented as a tuple of the users GUID and the custodians GUID. Further on a Home Location Register (HLR) based in the internet is responsible for the custodian-lookup of a user in case the custodian GUID is unknown. While the custodian node keeps track of its users, the HLR needs to be updated only if a user moves *and* changes his custodian. This hand-over is not trivial and may lead to lost or duplicate bundles.

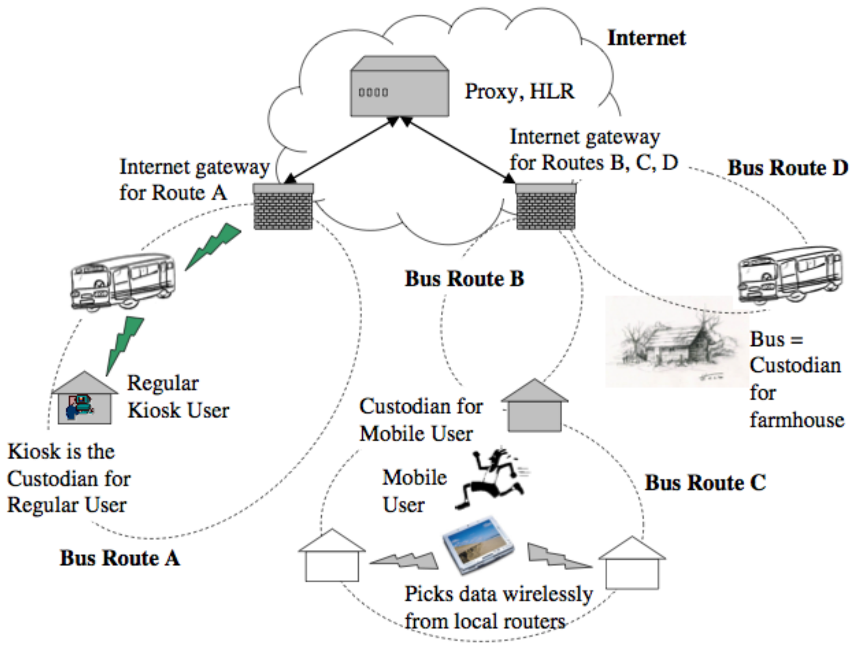
Internet
Proxy, HLR
Internet gateway for Route A
Internet gateway for Routes B, C, D
Bus Route D
Bus Route B
Bus = Custodian for farmhouse
Regular Kiosk User
Kiosk is the Custodian for Regular User
Custodian for Mobile User
Mobile User
Bus Route C
Bus Route A
Picks data wirelessly from local routers

Figure 5: The future DakNet needs to accomplish many different scenarios.

Routing

Three routing approaches are planed to be implemented, although none of them is revolutionary for the attentive reader. While *flooding* works fairly well in smallest networks, with its well known limitations, the *Reverse Path Forwarding* is a more efficient and economic approach. This *single copy* forwarding can be used after a initial *register* message has found the way across the custodian to the default internet gateway. Because every DTN node on the path registers the previous hop of the message, data to be sent to the user can easily follow the reverse path. One big advantage of *Reverse Path Forwarding* is that it allows bundles to take shortcuts. If a router along the path to the internet gateway has seen a *register* message recently, it can reroute the bundle to follow its own path. However, if a single node on the path fails, the protocol cannot recover itself and the bundle will be dropped. Timed rediscoveries can limit the outages to a maximum possible duration. Another handicap are the manually configured default routes to the internet gateway which can lead to suboptimal performance. The third routing protocol is based on *link state routing* and broadcasts periodically link state packets which allow to construct a weighted network topology graph. Beside the massive load because of the update packets, it seems to be difficult to choose the appropriate link weights. Not only expected meeting time should be represented, but also the queue length and load at the internet gateways.

Other measures concerning security and application support have been taken, which will not be further discussed in this tutorial. Many applications though have been 'ported' to this DTN system including email, FTP, HTTP-get etc. and can be used through a user transparent proxy installed at the kiosks and internet gateways.

Status

While the original DakNet has been successfully deployed in very few remote villages, this version intends to support about 80 kiosk and 5000 users. At this moderate scale *reverse path forwarding* seems to be appropriate, the Indian government, however, has announced to deploy up to 100'000 kiosks by the end of the year 2008. Clearly, the flat routing topology will not scale well with respect to the number of update messages. As the authors suggest, breaking up the topology into autonomous regions which are reachable through *border gateways* may be a possible solution for this problem.

5.2 ZebraNet

The ZebraNet [24] caused a little sensation and became known across the borders of DTN research (as a matter of fact, with some acceptable delay). In January 2004 multiple zebras have been stuffed with collars around their neck to collect data about their habits and behavior [18]. The Sweet-water game reserve in Kenya covers a area of about 100 square km and in the first release only seven out of the intended 30 collars have been deployed. Without going into details of a zebras social behavior, its movement pattern can be defined through the following states: *sleeping*, *grazing*, *graze-walking* and *fast-walking*. Also zebras head for water sources once a day and once there, drink relatively quickly. Some data of former biological studies has been used to simulate and optimize the ZebraNet protocol before deployment.

Protocol Design

A collar is equipped with a GPS device, flash RAM, a CPU and a separate short *and* long range radio. Estimated 30 position samples are taken per hour, 24 hours a day. The acquired data is transfered to other zebras within range at predefined time slots (total of 6 hours a day, using the short range radio) or delivered to the mobile sink node (total of 3 hours a day, using the long range radio). The mobile sink, represented in this case by an jeep, fully packed with curious zebra biologists, drives across the reserve for 3 hours a day and collects the data. Forwarding follows a simple but effective history-based scheme: Every collar initializes with a *hierarchy level* of zero. If it is able to connect

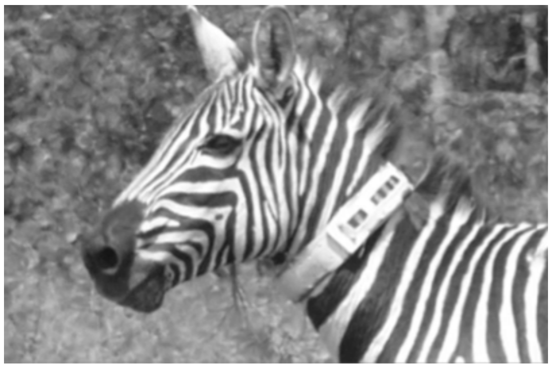
Figure 6: A zebra equipped with a ZebraNet collar.

to the mobile sink, the value is incremented, else it is decremented. This level indicates the 'closeness' to the base station and is a relatively reliable metric, unless the network topology changes very dynamically. Forwarding occurs to the neighbor with the highest hierarchy level, with ties randomly broken. In all transfers, the nodes send their own data first before forwarding other collars' data. As a last point, *deletion* of data in the buffer follows the rule to delete the oldest external data first and in case more space is required, the oldest internal (own) data. As soon as some messages have been delivered to the base station, a *delete list* is returned to the collar. This list is forwarded to other zebra peers as well and allows safe deletion of already delivered data.

Deployment Experiences

After a few days the zebras got used to their new decoration and data could be collected as planned. Beside some problems directing the collar upwards for good GPS signal reception and sun-rays to recharge the battery, the biggest technical issue was the limited radio range. Instead of 5 miles, the collars' radio range was only about 0.1 to 0.8 miles, which might have been caused as well by the zebras grazing position close to the ground. Some improvement needs to be done with the very restrictive duty cycles in which nodes can send and receive data. Future work should implement a more opportunistic scheme to allow devices to detect each other and communicate whenever they are within range.

6 Conclusion & Outlook

As seen in the previous chapters, DTN routing is and remains an interesting topic for further research. Many ideas have been brought onto paper and some concepts have proved, that they can be efficient and well performing under DTN assumptions. In the early days of DTN research, many simple heuristics have been tested and implemented, but they rarely lead to successful and fast routing (see for example the *MoVe* protocol). Within last years, research has done more fundamental and analytical work and evolved to an independent research field with a growing number of scientist related to these issues. Especially stateless algorithm concepts, which are often based on formerly known concepts and have been adapted to and optimized for DTN requirements, build a solid base for further DTN development. In general, the algorithms became more complex and comprehensive and this trend will surely continue within the next years.

While the main focus remains on the routing issue, other problem domains like *reliable data transfers*, *addressing schemes* or *security* need to be explored more in detail. Many actual approaches suffer of their specific and narrow focus on one issue, while to some extend fairly far fetched and unrealistic assumptions about mobility, buffer capacities, nodal behavior etc. intensify this effect.

One might wish to meet more analysis about basic, stateless mechanisms within a DTN in the future. Together with other approaches of different routing classes (i.e. history- or location-based algorithms) protocols will surely offer more efficiency in order of throughput, delay and overhead, and provide robust and scalable routing in such challenging networks. An other point of interest are more realistic and adequate mobility models, which are ideally extracted from field research and allow more insight into movement patterns to be used in (future) routing protocols.

Bibliography

[1] Evan P. C. Jones, Liliy Li, Paul A. S. Ward. *Practical Routing in Delay-Tolerant Networks.* SIGCOMM Workshops 2005.

[2] Sushant Jain, Michael Demmer, Rabin Patra, Kevin Fall. *Using Redundancy to Cope with Failures in a Delay Tolerant Network.* SIGCOMM 2005.

[3] Khaled A. Harras, Kevin C. Almeroth. *Inter-Regional Messenger Scheduling in Delay Tolerant Mobile Networks.* IEEE World of Wireless, Mobile and Multimedia Networks 2006.

[4] Jörg Widmer, Jean-Yves Le Boudec. *Network Coding for Efficient Communication in Extreme Networks.* SIGCOMM Workshops 2005.

[5] Yu Wang, Hongyi Wu. *DTF-MSN: The Delay/Fault-Tolerant Mobile Sensor Network for Pervasive Information Gathering.* IEEE 2006.

[6] A. Seth, S. Keshav. *Low-Cost Communication for Rural Internet Kiosks Using Mechanical Backhauls.* Proc ACM Mobicom 2006.

[7] Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, Hao-hua Chu. *A Hybrid Routing Approach for Opportunistic Networks.* SIGCOMM Workshops 2006.

[8] Jérémie Leguay, Anders Lindgren, James Scott, Timur Friedman, Jon Crowcroft. *Opportunistic Content Distribution in a Urban Setting.* SIGCOMM Workshops 2006.

[9] Brendan Burns, Oliver Brock, Brian Neil Levine. *MV Routing and Capacity Building in Disruption Tolerant Networks.* Proc INFOCOMM 2005.

[10] Jérémie Leguay, Timur Friedman, Vania Conan. *DTN Routing in a Mobility Pattern Space.* SIGCOMM 2005.

[11] John Burgess, Brian Gallagher, David Jensen, Brian Neil Levine. *MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks.* Proc INFOCOMM 2006.

[12] Thrasyvoulos Spyropoulus, Konstantinos Psounis, Cauligi S. Raghavendra. *Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks.* SIGCOMM Workshops 2005.

[13] Tara Small, Zygmunt J. Haas. *Resource and Performance Tradeoffs in Delay-Tolerant Wireless Networks.* SIGCOMM Workshops 2005.

[14] Matthias Grossglauser, Martin Vetterli. *Locating Nodes with EASE: Last Encounter Routing in Ad Hoc Networks through Mobility Diffusion.* INFOCOM 2003.

[15] Joy Gosh, Seokhoon Yoon, Hung Ngo, Chunming Qiao. *Sociological Orbits for Efficient Routing in Intermittently Connected Mobile Ad Hoc Networks.* UB CSE Tech Report 2005.

[16] Jason LeBrun, Chen-Nee Chuah, Dipak Ghosal. *Knowledge-Based Opportunistic Forwarding in Vehicular Wireless Ad Hoc Networks.* IEEE VTC 2005.

[17] Xiangchuan Chen, Amy L. Murphy. *Enabling Disconnected Transitive Communication in Mobile Ad Hoc Networks.* POMC 2001.

[18] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, Margaret Martonosi. *Hardware Design Experiences in ZebraNet.* SenSys 2004.

[19] Amin Vahdat, David Becker. *Epidemic Routing for Partially-Connected Ad Hoc Networks.* Duke University Technical Report 2000.

[20] Michael Demmer, Eric Brewer, Kevin Fall, Sushant Jain, Melissa Ho, Rabin Patra. *Implementing Delay Tolerant Networking.* Intel Corporation 2004.

[21] Sushant Jain, Kevin Fall, Rabin Patra. *Routing in Delay Tolerant Network.* SIGCOMM 2004.

[22] Forrest Warthman. *Delay-Tolerant Networks. A Tutorial.* DTN Research Group Internet Draft 2003.

[23] Alex Pentland, Richard Fletcher, Amir Hasson. *DakNet: Rethinking Connectivity in Developing Nations.* IEEE Publication 2004.

[24] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, Daniel Rubinstein. *Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet* ASPLOS 2002.

[25] V. Cerf, S. Burleigh et al. *Delay-Tolerant Network Architecture* RFC 4838.

[26] G. J. Alexander, J. C. Francis. *Portfolio Analysis.* Prentice Hall, 1986.

[27] M. Savelsbergh. *Local search in routing problems with time windows.* Annals of Operations Research, 1985.

[28] Douglas Adams. *A Hitchhikers Guide to Galaxy.* BBC Radio 4, 1978.