PLANNING AND LEARNING ALGORITHMS FOR ROUTING IN DISRUPTION-TOLERANT NETWORKS

Mark-Oliver Stehr and Carolyn Talcott
SRI International, Computer Science Laboratory, Menlo Park, California 94025, USA

***Abstract*—*We give an overview of algorithms that we have been developing in the DARPA Disruption-Tolerant Networking program, which aims at improving communication in networks with intermittent and episodic connectivity. Thanks to the use of network caching, this can be accomplished without the need for a simultaneous end-to-end path that is required by traditional Internet and mobile ad-hoc network (MANET) protocols. We employ a disciplined two-level approach that clearly distinguishes the dissemination of application content from the dissemination of network-related knowledge, each of which can be supported by different algorithms. Specifically, we present probabilisitc reflection, a single-message protocol enabling the dissemination of knowledge in strongly disrupted networks. For content dissemination, we present two approaches, namely a symbolic planning algorithm that exploits partially predictable temporal behavior, and a distributed and disruption-tolerant reinforcement learning algorithm that takes into account feedback about past performance.*

I. DELAY/DISRUPTION-TOLERANT NETWORKING

In the current generation of Internet protocols based on the packet switching paradigm, packets are discarded by routers immediately after forwarding, independent of their sucessful reception or acceptence at the next hop. Essentially, this means that sucessful delivery of a packet requires the existence of a sufficiently stable end-to-end path at some instant in time. However, in today's networks, which are often characterized by intermittent connectivity, there are various reasons why maintaining such paths becomes impossible or even undesirable. For instance, node failures and disruptions are part of the normal operation in mobile ad hoc networks set up for military operations or emergency response. Space-ground-networks (e.g. involving satellites) are often characterized by episodic connectivity and long latencies. In many applications, e.g. in sensor networks, resource constraints (e.g. regarding energy and spectrum) make it undesirable to keep active all nodes and links all the time even if it would be physically possible. More generally, there is need to support communication even in scenarios when end-to-end paths never exists, such as in networks relying on message ferries, e.g. vehicles, unmanned aerial vehicles (UAVs), or simply personal digital assistants (PDAs), for the transport of data. The trend of an explosively growing number of energy-constrained networked devices competing for communication spectrum growing at a much slower pace indicates that intermittent connectivity will become an even more common mode of operation in the near future. However, even in cases where the probablility of stable end-to-end paths is reasonably high, end-to-end retransmission such as implemented by TCP/IP may not be a satisfactory solution, because first it relies on relatively small round-trip times, and second the waste of resources by retransmitting data that has made it half-way to its destination is considerable and usually occurs when the network is already stressed enough.

The goal of delay/disruption-tolerant networking (DTN) [7] is to design protocols that are suitable for networks with intermittent connectivity of various degrees of predictability, ranging from unpredictable networks, arising from random movements of people or vehicles, to highly predictable networks, e.g. satellite networks based on well-defined satellite orbits. To overcome disconnection, content will be stored in the network for extended periods of time up to an expiration time that can be specified for each unit of information, which is called a bundle in DTN terminology. Since the use of storage is so central to the solution, we think of DTN, more generally, as an experiment of what can be achieved in storage-rich networks, i.e. if storage would not be a constraining resource, an assumption fundamentally different from that underlying the Internet protocols. The justification for being interested in this case is that a rapidly growing gap can be observed between the cost of storage vs. the cost of communication bandwidth making storage relatively less expensive and storage-rich ad hoc networks an increasingly relevant case.

II. RELATED WORK

The Delay-Tolerant Networking Research Group (DTNRG) proposed the delay-tolerant network architecture [6], [2] based on a store-and-forward paradigm and a notion of custody transfer, which we more generally view as a special case of network caching. Our experimental work is based on the DTN reference implementation that was originally developed by the DTNRG and further advanced

We gratefully acknowledge support from DARPA through Contract W15P7T-06-C-P427. Distribution Statement A: Approved for Public Release, Distribution Unlimited.

978-1-4244-2677-5/08/$25.00 ©2008 IEEE

in the DARPA DTN program [11] towards an architecture for policy-based networking using a knowledge-based, declarative approach.

Jain et al. [10] described a number of routing approaches in delay-tolerant networks. These approaches are based on oracles to provide nodes with complete or partial information about contacts, queuing at nodes, and traffic demands. The novelty of the overall approach lies in the consideration of the temporal aspect of network connectivity. The limitations of these approaches are that they focus on single-path routing, and are based on modifications of Dijkstra's shortest-path algorithm, which relies on a relatively up-to-date view of the network topology.

Another effort that considers the temporal aspect of network connectivity is the *space-time routing (STR)* approach proposed in [5]. STR consists of an on-demand routing scheme based on distances to destinations that incorporates information about the times when routes to destinations are established and the times when packets are originated by their sources. However, the inclusion of the temporal dimension in STR is limited to the freshness of information.

For capturing the dependency of routing on space and time, [15] proposes a different framework which is suitable for networks with predictable mobility and attempts to capture the time dependency of links rather than the freshness of information as STR does. The new framework is based on space-time graphs, which are layered graphs with each layer corresponding to a discrete time interval in the lifetime of the network and containing all network nodes.

Epidemic routing has been proposed for partially connected networks when nodes move randomly. The approach by [20] uses summary vectors stating the messages that have been received by nodes, and packets are forwarded to all neighbors. Serveral ways to control the message flooding of epidemic routing, see e.g. [13] or [17], have been developed to overcome its high overhead, including *directed diffusion* [9] which has been proposed as a scalable and robust communication paradigm for sensor networks. It refers to a class of routing algorithms based on interest and reinforcement learning. Their objective is to route streams of small units of information from sources to sinks guided by interest. Directed diffusion is based on selecting paths between source and sink with the idea that the choice can adapt to changing network conditions. For backward reinforcement and suppression of paths, there is an assumption that routers have some freedom to locally adapt the rate (e.g. by omitting sensor readings), which is not an option for the potentially large bundles that DTNs are concerned with. In contrast to bundles in DTNs, units of information in directed diffusion are typically short-lived, and although they are stored in local data caches, the purpose of caching is the avoidence of routing loops rather than to overcome intermittent connectivity or increase information availability. Hence, interest attracts streams of data from all sources rather than locating one replica of a bundle that is stored in the network independent of when and where it originated.

A few algorithms for learning-based routing have been developed, although all of them in the context of an IP-style rather than a DTN-style model. An early reference is [1], which proposes *Q-routing*, a reinforcement learning approach to packet routing in dynamically changing networks. This work presents a simple routing algorithm that adjusts the weight at each router for choosing a certain outgoing link according to historical effectiveness. Q-routing has several drawbacks: it is deterministic and does not allow exploratory behavior so that some links may never be tried, network changes may not be discovered, and the well-known counting-to-infinity effect can occur. Related approaches are *Predictive Q-routing* [3] and *DRQ-routing* [12]. Another class that has attracted some interest is that of ant routing algorithms (see e.g. [8]), which are also based on reinforcement learning.

A distributed learning approach that was originally designed for robots but has been applied to routing algorithms is presented in [18]. It is named TPOT-RL for *Team-Partitioned, Opaque-Transition Reinforcement Learning*, and is a distributed reinforcement learning technique that allows a team of independent agents to learn a collaborative task. According to the published empirical results, individual nodes can learn to efficiently route packets through a network that exhibits changing traffic patterns based on local sensing. Unfortunately, this algorithm suffers from weaknesses similar to those of Q-routing, except that counting-to-infinity can be avoided. A quite different approach to multiagent learning that avoids these problems is routing based on OLPOMDP [19], a policy-gradient reinforcement learning algorithm.

Peer-to-peer networking [14] and (adaptive) web caching [21], [16] and are research areas that have a similar emphasis on replication and caching as DTN. In fact, their top level objective is the same, namely the improvement of information availability and avoidence of costly end-to-end retransmissions. However, nearly all current systems, a notable exception being the approach to peer-to-peer file-sharing proposed in [4], are designed for fixed infrastructure rather than for mobile ad hoc or even disruption-tolerant networks. Their algorithms usually abstract to a large degree from the underlying physical network topology and the routing process.

In our view the interesting feature of DTN is that both caching and routing are considered on an equal footing.

III. Technical Approach

In order to function properly, the network nodes need to operate in a reasonably coordinated way, but what is needed is a very loosely coupled communication paragdigm that supports local autonomy without requiring peers to wait for each other before being able to proceed with the local operations.

Since knowledge about the network is the basis for decisions in our approach, this is achieved by evolving the network towards a state where all nodes have sufficient knowledge to make local decisions and where the distributed knowledge is approximately consistent. We acknowledge that due to delays and bandwidth limitations neither full consistency nor completeness of the knowledge distribution can be achieved, and design our routing algorithms to tolerate partial and inconsistent knowledge.

The approach to disseminate certain kinds of knowledge is present in many traditional routing protocols such as link-state routing, where link-state advertisements are disseminated, or distance-vector routing, where knowledge about distance to potential destinations is exchanged. In both cases the aggregated knowledge is later used as a basis for routing decisions. Our approach is generic in the sense that to a large degree it abstracts from the particular kind of knowledge that is disseminated and also allows the choice between different knowledge dissemination protocols.

Since the ultimate goal of traditional networking is to disseminate application content, we clearly distinguish knowledge (about the network) from content (application payload). This separation also reflects fundamental differences in their distributed processing. Knowledge is subject to in-network processing, whereas content is stored and forwarded logically without modification. Knowledge is fed into the local knowledge base at each node and can be subject to deletion and aggregation.

The notion of a path between source and destination is not very useful in DTN, not only due to intermittent connectivity, but because also publisher and subscriber of bundles can be temporarily decoupled. In DTNs we distinguish a node identifier that uniquely identifies a single node, e.g. the source of a bundle, from a symbolic address expression, e.g. the destination of a bundle, that is a symbolic representation of a set of nodes. The interpretation of a symbolic address is a function of time, because it represents the set of nodes registered/subscribing to this address, which can significantly change over the lifetime of a bundle. Hence, DTNs use late binding, that is the delayed binding of nodes to symbolic addresses, as opposed to e.g. the resolution of addresses at the source via the domain name system (DNS).

Symbolic addresses allow for various interpretations, e.g. a symbolic address can stand for an individual, a group, a function, a role, or a topic. Depending on the expressivity of the language, the user can form symbolic address expressions that can be matched (or logically speaking, satisfied) by a set of symbolic addresses. For instance, our implementation in the DTNRG framework currently supports symbolic address destinations containing wildcards (logically speaking, existential variables), but our routing algorithms are independent of the syntax and semantics and hence would work equally well with more expressive languages or logics.

A. Knowledge Representation and Dissemination

The local knowledge base is a set of knowledge items conveying the status and statistics about various entities. Each knowledge item has at least the following attributes associated with it: the *creator*, i.e. the node that created it; its *creation time* and its *expiration time*.

All node-related knowledge items contain information about the node that is the creator of the knowledge item. All link-related knowledge items have an additional *link* attribute that relative to the creator uniquely identifies an outgoing link. An interest knowledge item has an additional *address* attribute, which is a set of symbolic addresses that the creator is registered for. A bundle-delivered status knowledge item has a *bundle* attribute that uniquely identifies an (end-to-end) bundle. A bundle-received status knowledge item has a *bundle* attribute that uniquely defines a bundle fragment (a complete bundle being a special case) in the network.

In the following, we define an equivalence relation $\equiv$ and a partial ordering $\prec$ on knowledge items. Intuitively, the equivalence $k \equiv k'$ means that k and k' contain the same information, and the relation $k \prec k'$ means that k' subsumes k. In the latter case we also say that k is obsolete given k' or that k' is fresher that k. For the operation of the knowledge manager, $k \prec k'$ implies that if k is in the local knowledge base and k' is received then k should be replaced by k'. Similarly, $k \equiv k'$ implies that if k is in the local knowledge base and k' is received then k' is redundant and can be ignored.

The main protocol that we have developed for knowledge dissemination is called *probabilistic reflection*. For comparison purposes, we have also implemented two simple flooding algorithms that we call *deterministic flooding* and *periodic advertisement*.

Both deterministic flooding and periodic advertisement have their drawbacks in the sense that the former disseminates knowledge not sufficiently often (only once) to overcome failures whereas the latter disseminates the same knowledge again and again without taking into account

any feedback. Hence it is natural to try to find a solution which combines the advantages of both with an overhead that is somewhere in the middle. A clear advantage of both protocols is that they both do not use complicated (multiround) protocols. In fact, not even a round trip, but a single message transmission is sufficient to make progress, which means that they can utilize the smallest windows of opportunity. This is in contrast to more complex epidemic protocols which first require messages, exchanging so-called summary vectors, before exchanging the actual knowledge, and hence require bidirectional links and a larger window of opportunity.

Probabilistic reflection makes minimal assumptions on the network as well. Links can be bidirectional or unidirectional, since feedback does not have to be direct. The error rate can be high. Complete dissemination of knowledge is guaranteed under the eventual weak connectivity assumption. It is a single message protocol with medium overhead, because a knowledge item is possibly disseminated more than once, but higher numbers of retransmissions occur with decreasing probability.

Probabilistic reflection makes use of knowledge about knowledge to reduce the number of unnecessary retransmissions. To this end, we keep an additional *awareness* attribute for each knowledge item, which is the set of nodes that, according to the local knowledge, are already aware of this item. If a knowledge item k' is received and there is an existing knowledge item k in the local knowledge base such that $k \equiv k'$, then k will not be replaced by k' but the awareness set of k will be extended by the awareness set of k'. In this way awareness propagates though the network even if no fresh knowledge is generated. The awareness set is now taken into account to eliminate redundant transmissions of knowledge items. A knowledge item is only transmitted if the receiving neighbor is not already in its awareness set.

Specifically, each node participating in probabilistic reflection executes the following algorithm. Periodically, for each new knowledge item k and for each each outgoing neighbor one of the following actions is performed: (1) If the potential receiver is not known to be aware of k, then k will be sent out on the corresponding link. (2) Otherwise, k is sent out with a non-zero probability $p = r/n$ where r is a reflection parameter and n is the number of outgoing neighbors.

Intuitively, a large reflection parameter leads to unnecessary high overhead due to repeated reflection of knowledge in the network. A small reflection parameter, on the other hand, prevents feedback to populate awareness sets and hence also leads to frequent retransmissions of knowledge items. For a wide range of network conditions and parameters between 0.6 and 0.9, however, we have found that probabilisitic reflection provides relatively low overhead and a good compromise between flooding and periodic advertisement. Due to space limitations, we will not atempt a detailed performance comparison of knowledge dissemination algorithms in this paper, but instead we focus on the performance of content dissemination using deterministic flooding as an underlying knowledge dissemination protocol, because it is sufficient for sample scenario.

B. Content Dissemination and Caching

We have developed three approaches to content dissemination which all use *opportunistic content replication and caching* to store and hence replicate bundles on the way to the destination whenever possible. Altough more proactive approaches are possible, we are currently focussing on the opportunistic approach, since the networking overhead added relative to routing without caching is zero. The issue of cache management can be separated from the replication strategy. Currently, we use a *lazy deletion policy* that stores bundles as long as possible and discards bundles only if the expiration time has been reached.

1) Route Planning: Our route planning is reflective in the sense that it uses a perceived model of the network reality as the basis for the planning process. To this end, the delayed approximation of the current known network conditions, which is given by the state of the local knowledge base, is translated into an internal symbolic network model that can be executed using reflective techniques. The goal of route planning is to produce a content dissemination plan, which is then attached to the bundle and executed while the bundle is in transit. Plans are not necessarily linear, as for instance source routes, but have a branching structure to account for alternatives and concurrent forwarding. Although our DTN simulator has been developed in Java, we used the rewriting logic language Maude in the development of the reflective route planner, because a logic-based language is better suited to deal with constraints, reflection, and the planning process.

Borrowing ideas from Petri net theory, a *content dissemination plan* is a directed acyclic graph (representing a causal order) with two different kinds of nodes. Each edge represents the forwarding of a bundle to a neigbor, and the entire graph explains how a bundle can be routed from a source to a destination, allowing for planned multi-copy routing, and planned multi-choice routing. The two kinds of nodes are AND nodes and OR nodes. AND nodes express multi-copy routing, meaning that the bundles needs to be routed concurrently. OR nodes express routing alternatives, e.g. rerouting on failure. The alternatives are explored in

a particular order (specified by annotations), meaning that the first applicable alternative will be chosen.

The planner consists of three phases: building a network model, searching for feasible routes (linear plans), and assembling a set of routes into a plan. The external interface for the planner is a function that takes a number of parameters specifying the planning problem - a knowledge base, source and destinations nodes, bundle size - and additional parameters that control the planning process. It first computes an executable network model, reflects it to the meta-level, and applies a state space exploration function to the reflected model and a suitably constructed initial state. The result of the search is a set of feasible traces, that we interpret as linear plans. Different from link-state routing, route planning does not require up-to-date link-state information, but can utilize delayed information about (time-stamped) link-state events and link statistics to evaluate the quality of each linear plan. This set of linear plans is then fed into a function along with the parameters specifying the plan selection and composition process. The main options are to synthesize a single best linear plan or a branching plan with parameters to control branching and the ordering of choices in multi-choice branches.

2) Interest-Driven Routing: Our interest-driven routing algorithm is designed to support multiparty communication based on the idea of active interest signaling from interested destinations. It uses the knowledge about interest, i.e. registrations for a symbolic address, which is disseminated by the underlying distributed knowledge manager between neighbors. The direction from which interest in a given symbolic destination is first received is used to estimate the best routing path for bundles toward the corresponding symbolic address. In general, however, interest can originate from multiple interested parties, which are situated in different directions from the viewpoint of the current router. Hence, if interest for a bundle comes from multiple directions, the bundle is forwarded simultaneously over multiple corresponding links.

To support interest-driven routing, the knowledge manager maintains a *received-from* attribute for interest knowledge items, which is a list of nodes. If the knowledge manager receives a knowledge item k from a neighbor n there are two cases. If k is fresh the knowledge manager creates a local copy with the received-from attribute set to the singleton list containing n. Otherwise, there is a unique knowledge item k' in the local knowledge base such that $k \prec k'$, and the knowledge manager adds n at the end of the received-from list of k' if n is not already in the list.

When processing a bunde b with destination d, the decision logic of interest-based routing uses this information as follows. For all known nodes n of the network (this information is available as node knowledge in the local knowledge base) and for all outgoing links l that are ready for transmission or retransmission of bundle b, perform the following: If a bundle interest with creator n and address d exists in the local knowledge base and its received-from list has the receiver r of link l as its first element then b will be transmitted to r. As an optimization we ignore nodes n in the above procedure for which it can be determined (based on the local bundle status knowledge) that have already received b.

3) Learning-Based Routing: Our learning-based routing algorithm is based on distributed reinforcement learning, which adapts its routing policy in response to the rewards it receives from delivered bundles. A reward function of is used to assess the quality of a routing decision. In the case of a single destination this function is simply $1/\Delta$ where Δ is the transit time of the data bundle. Hence, in the limit case, a bundle that is not delivered will not lead to an increase of reward. Punishments or negative rewards do not exist in our approach. The function is then generalized to the case of multiple destinations by adding i/Δ_i for each additional node $i \geq 2$ at which the bundle has been delivered after a transit time of Δ_i. In this way, the number of destination nodes can remain open-ended and there is a strong incentive to deliver to as many nodes as possible.

Two challenges of using a reinforement learning approach in DTN routing lie in the potentially large action space and the need to operate in a highly distributed and delay-/disruption-tolerant fashion without strong synchronization assumptions. A further challange is to associate causes and effects in such an environment so that the reward can be credited to agents and actions that have contributed to the succesful delivery of a bundle to its destination. The latter is known in the literature as the credit assignment problem, but it is more complicated in our context, because even the question whether (or more precisely to what degree) the objective has been achieved cannot be decided locally.

a) Selection of Forwarding Actions: Like our interest-driven routing algorithm, learning-based routing supports singleton and non-singleton destinations. This is achieved by associating with each node a set of forwarding actions, where each forwarding action is defined by a subset of its neighbors to each of which a node transmits a bundle if this action is performed. To limit the complexity of the action space we usually constrain the cardinality of forwarding actions, and hence the maximum number of neighbors a bundle can be transmitted to in a single routing decision.

Our learning-based routing algorithm consists of *exploration* and *exploitation* subalgorithms. The choice of the algorithm is determined by a Bernoulli experiment for each

bundle that needs to be routed or reconsidered for routing, e.g. due to the need for a retransmission.

The difference between exploration and exploitation lies in the way the forwarding action is selected. Although it may sound counterintuitive, a forwarding action will always at least contain the previous hop of an incoming bundle, because in this way valuable information about the context of the decision can be represented as part of the action. Another possible picture consistent with this idea is to view an action as a local undirected abstraction of the bundle flow, which contains both the immediate source and the immediate destinations of a bundle. In the case of *exploration*, we essentially use the (unoptimized) interest-driven routing algorithm to compute the forwarding action. The idea behind *exploitation* is to exploit learned knowledge (that has been accumulated in terms of rewards) to make the best possible routing decision based on a window of recorded forwarding actions (in the so-called forwarding action log) performed in the recent past. Our current algorithm deterministically selects the best possible forwarding action, i.e. the one that has received the maximal average reward, with ties broken probabilistically.

Once the forwarding action is computed, the current bundle can be sent to all neighbors contained in this action. The optimization of interest-driven routing that interested nodes that are known to have received the content already is not used in the computation of the forwarding action, because learned actions should remain as general as possible. Similar to interest-driven routing, a bundle is never sent back to the previous hop and, furthermore, bundle status knowledge is used to avoid uneccessary transmissions of bundles to neighbors which are known to have received the bundle already.

b) Credit Assignment and Reward Computation: The computation of reward is performed whenever fresh bundle-delivered knowledge is received. There are two levels of computations. First, based on the above-mentioned reward function, a reward is added for each recorded action which *may* have contributed to the sucessful delivery of the bundle, meaning that the transit time is a positive number that is not smaller than some configurable lower bound. Second, for each symbolic destination and each possible action of the node we compute the corresponding entry in the average reward matrix by averaging the rewards for all recorded actions that correspond to this entry. Only the reward matrix is used for routing decisions in the exploitation subalgorithm. Since recorded actions are removed from the forwarding action log after some configurable expiration time (usually larger than the expiration time of bundles) the recorded actions and hence the reward matrix represents the cumulative reward for a window of past behviour.

C. Performance in a Typical Scenario

Most existing network simulators have been specifically developed to study protocols for the traditional IP stack and hence rely on very specific and detailed models of all layers. However, at the current stage our objective is to evaluate and compare different DTN algorithms independent of the underlying networking technology. To this end, we have developed a DTN network simulator that abstracts from the underlying networking stack in order to obtain approximate performance results. It uses use a simple graph-based pipe model of potential links (which can be up or down) where each link is characterized by its abstract features such as bandwidth, latency, error rate, and the distribution of disruptions parameterized by average up and down times.

Network dynamics of mobile ad hoc networks is often generated by traditional simulators using very specific physcial models of mobility, e.g. random waypoint, which are often based on simplistic assumptions about the movement of nodes. In our analysis, we allow the specification of abstract mobility models for a concrete scenario, where instead of actual coordinates we use regions defining equivalence classes of states or locations exhibiting similar connectivity. The battlefield scenario in Fig. 1 is an example of such an abstract dynamic model. As an alternative to specific scenarios, we study models generated from a well-defined family of synthetic models using predefined distributions. By averaging the performance metrics over a large number of such models we obtain a more realistic expectation of the range of results and can reduce the likelihood of overfitting our algorithms to specific scenarios.

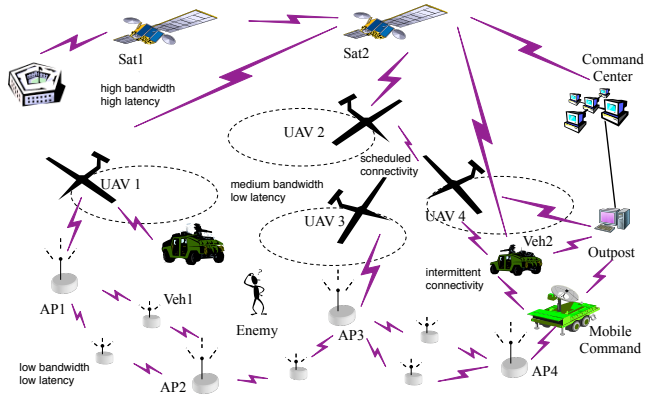

Fig. 1. Battlefield Scenario

Table I shows how the delivery time varies with the choice of the algorithm in battlefield scenario with traffic between random sources and destinations. The results have been obtained by averaging 100 simulation runs (10 in case of planning). To enable a more precise understanding of the performance of our algorithms we have disabled the

retransmission of content bundles, which would eventually lead to a 100% delivery rate. Hence, the delivery rates can be interpreted as delivery rates for the first attempt.

Not surprisingly, optimized flooding shows a high first-attempt delivery rate, but has the drawback of a high network load, which in turn leads to high delivery times. Flooding is already optimized in the sense that, like all our routing algorithms, it exploits knowledge about the reception of bundles to avoid redundant transmissions. Shortest path routing computes the shortest path in the potential link graph and hence leads to uneccessary long delays due bundles waiting for specific links to open. Link-state routing needs to disseminate the link-state but can significantly improve the delivery time by chosing a shortest path in the subgraph of open links. The planning approaches can improve the delivery times or rates by exploiting temporal predictions and multi-path routing. In general, our planning algorithms can operate with a reduced overhead by relying on less frequently disseminated link statistics instead of current link-state information. The performance of interest-driven routing is similar to link-state routing, but it has lower overhead because the dissemination of link-state information is entirely avoided.

Table I. Simulated Performance

	Delivery Rate	Utilization	Number of Hops	Delivery Time
Optimized Flooding	87.28 %	0.97 %	3.12	13.97s
Shortest Path Routing	76.77 %	0.07 %	2.30	12.42s
Single-Path Planning	80.56 %	0.14 %	2.68	6.53s
Multi-Choice Planning	79.92 %	0.14 %	2.81	6.16s
Multi-Copy Planning	94.17 %	0.23 %	2.77	8.18s
Link-State Routing	79.77 %	0.28 %	2.52	9.29s
Interest-Driven Routing	78.57 %	0.09 %	2.55	9.41s

To study the performance of learning-based routing relative to interest-driven routing, each simulation run is organized in two phases: In the first phase, the learning-based routing algorithm is executed purely in exploration mode, meaning that it behaves like interest-driven routing, but records all information learned from the feedback. In the second phase, the learning algrithm exploits all previously accumulated knowledge to make routing decisions.

From each such simulation we now extract two data points as depicted in Fig. 2. The first point is given by the network utilization (y-axis) and delivery rate (x-axis) of the interest-driven routing used for the Phase 1, and the second point is given by the corresponding results for Phase 2. Connecting these points by an arrow gives us a graphical visualization of the impact of learning in one particular run by means of a family of vectors. For a representative average, we have accumulated the vectors of 100 simulation runs. From the direction of the arrows, a general tendency

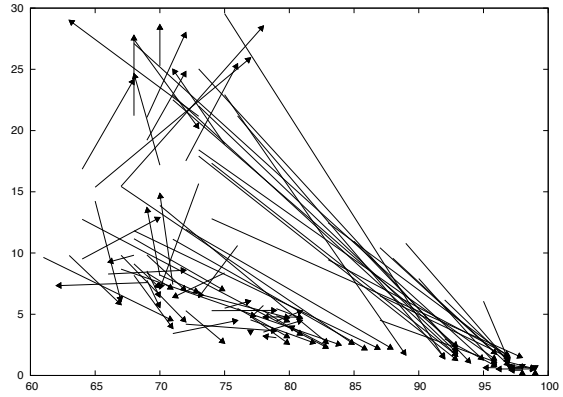

Fig. 2. Relative Performance of Learning-Based Routing

for arrows to improve both delivery time and delivery rates can be observed. Table II summarizes the averages over this set of simulations, and confirms that in average a factor of about 1.6 in reduction of delivery time together with a modest improvement in the delivery rate can be achieved.

Table II. Relative Performance in Average

	Delivery Rate	Utilization	Number of Hops	Delivery Time
Exploration	78.48 %	0.09 %	2.39	9.79s
Exploitation	84.10 %	0.09 %	2.44	6.06s

IV. Conclusions

We have introduced three routing algorithms for content dissemination in disruption-tolerant networks, namely interest-driven routing, route planning, and learning-based routing, and studied their performance relative to three simple baseline routing algorithms, namely optimized flooding, shortest-path routing and link-state routing. We have evaluated our algorithms under a variety of network topologies and conditions, but due to space limitations we have only presented selected results for a particular scenario in this paper. We have demonstrated that our interest-driven routing algorithm can yield average delivery times lower than (shortest-path) link-state routing with lower overhead thanks to the dissemination of interest instead of frequently changing link-state information. We have also shown that route planning can improve delivery times over (shortest-path) link-state routing by exploiting the predicted temporal evolution and can deal with unpredictability by planning for alternatives or redundant routes. Finally, we have investigated the effect of our newly developed learning-based routing algorithm, which uses a distributed reinforcement learning approach and utilizes interest-driven routing as a base algorithm to explore relevant routing opportunities with low overhead. Our simulations over a large number of different scenarios and network conditions show improvements in the delivery time of a factor between 1 and 2

for unstructured dense networks and over 4 for structured networks with exploitable asymmetries. We also found that that learning tends to increase the first-attempt delivery rates (and hence reduces the need for retransmissions) with no or modest increase of the network load. However, we have also found entire classes of scenarios, where learning does not show significant improvements or only improvements in the delivery rate while at the same time increasing the delivery time. The results for learning have to be interpreted in the light that the underlying interest-driven routing has been shown to work already remarkably well for the timely dissemination of information for a wide spectrum of network conditions. Furthermore, learning-based routing can only improve the performance if opportunities for improvement are offered by the underlying physical network.

Since disruption-tolerant networking aims at covering a wide range of possible applications, it is most likely that a better understanding of the tradeoffs between different approaches will eventually lead to a family of protocols with phase-transitions that determine which protocols should be used where and when. To complete the picture, further metrics need to be investigated, especially those relevant at the application level (e.g. QoS metrics for video streaming). One big challenge that remains is to develop a framework based on higher-level knowledge that could be expressed in the form of networking policies in which various protocols interoperate seamlessly and cooperatively. In our view, a knowledge-based approach is a good starting point, but a lot more work is needed on the semantic level of interaction, interoperation, and on the abstract specification of networks, algorithms and their capabilities.

Acknowledgements: We would like to thank Thomas Scharler for his help with the implementation of several of our algorithms in the DTN reference implementation, in particular probabilistic reflection, interest-driven routing, and learning-based routing; Sebastian Guteirrez-Nolasco for improving the simulation component; and Mike Demmer for fixing bugs and implementing several of our suggestions. We are grateful to our collaborators Ignacio Solis and J.J. Garcia-Luna-Aceves for the opportunity of using the wireless testbed at PARC Palo Alto Research Center. We also appreciate the interest and various contributions of our visiting students Gianluca Capuzzi and Egidio Cardinale.

REFERENCES

[1] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, volume 6, pages 671–678, 1993.

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

[3] S. P. M. Choi and D.-Y. Yeung. Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control. In *Advances in Neural Information Processing Systems*, volume 8, 1996.

[4] G. Ding and B. Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, page 104, Washington, DC, USA, 2004. IEEE Computer Society.

[5] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Space-time routing in ad hoc networks. In *Proceedings Ad Hoc Now 03, Montreal, Canada, October 2003*.

[6] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM, 2003.

[7] Stephen Farrell and Vinny Cahill. *Delay- and Disruption-Tolerant Networking*. Artech House Publishers, September 2006.

[8] Mesut Günes and Otto Spaniol. Ant-routing algorithm for mobile multi-hop ad-hoc network. In *Network Control and Engineering For QoS, Security and Mobility II*, pages 120–138. Kluwer Academic Publishers, 2003.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[10] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158. ACM, 2004.

[11] R. Krishnan, P. Basu, J. M. Mikkelson, C. Small, R. Ramanathan, D. Brown, J. Burgess, A. Caro, M. Condell, N. Goffee, R. R. Hain, R. Hansen, C. Jones, V. Kawadia, D. Mankins, B. Schwartz, T. Strayer, J. Ward, D. Wiggins, and S. Polit. The SPINDLE disruption-tolerant networking system. In *Proceedings of MILCOM 2007, Orlando, FL, November 2007*, 2007.

[12] S. Kumar and R. Miikkulainen. Dual reinforcement Q-routing: An on-line adaptive routing algorithm. In *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE-97, St. Louis, MO)*, volume 7 of *Smart Engineering Systems: Neural Networks, Fuzzy Logic, Data Mining, and Evolutionary Programming*, pages 231–238. ASME Press, 1997.

[13] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[14] Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[15] S. Merugu, M. Ammar, and E. Zegura. Routing in space and time in networks with predictable mobility. Technical Report GIT-CC-04-07, College of Computing, Georgia Institute of Technology, March 2004.

[16] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching: towards a new global caching architecture. *Comput. Netw. ISDN Syst.*, 30(22-23):2169–2177, 1998.

[17] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of SIGCOMM 2005*.

[18] P. Stone. TPOT-RL applied to network routing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 935–942, 2000.

[19] N. Tao, J. Baxter, and L. Weaver. A multi-agent policy-gradient approach to network routing. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 553–560, 2001.

[20] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000.

[21] J. Wang. A survey of web caching schemes for the internet. *SIGCOMM Comput. Commun. Rev.*, 29(5):36–46, 1999.