PreDA: Predicate Routing for DTN Architectures over MANET

Flavio Esposito   Ibrahim Matta

Department of Computer Science, Boston University
Boston, MA, USA
{flavio, matta}@cs.bu.edu

Abstract—**We consider a Delay Tolerant Network (DTN) whose users (nodes) are connected by an underlying Mobile Ad hoc Network (MANET) substrate. Users can declaratively express high-level policy constraints on how "content" should be routed. For example, content can be directed through an intermediary DTN node for the purposes of preprocessing, authentication, etc., or content from a malicious MANET node can be dropped. To support such content routing at the DTN level, we implement Predicate Routing [1] where high-level constraints of DTN nodes are mapped into low-level routing predicates within the MANET nodes. Our testbed [2] uses a Linux system architecture with *User Mode Linux* [3] to emulate every DTN node with a *DTN Reference Implementation* code [4]. In our initial architecture prototype, we use the On Demand Distance Vector (AODV) routing protocol at the MANET level. We use the network simulator *ns-2* (ns-emulation version) to simulate the wireless connectivity of both DTN and MANET nodes. Preliminary results show the efficient and correct operation of propagating routing predicates. For the application of content re-routing through an intermediary, as a side effect, results demonstrate the performance benefit of content re-routing that dynamically (on-demand) breaks the underlying end-to-end TCP connections into shorter-length TCP connections.**

I. INTRODUCTION

Motivation: Delay Tolerant Networks [5] include a vast class of challenged networks where, by their nature, communicating nodes would never or rarely have a stable end-to-end path. A significant subclass of such challenging networks is represented by Mobile Ad hoc Networks.

We consider a Delay Tolerant Network (DTN) whose users (nodes) are connected by an underlying Mobile Ad hoc Network (MANET) substrate, and are provided with the capability of declaratively specifying constraints on routing content. Although declarative approaches have been widely discussed as clean-slate alternative to routing [6] or transport protocols [7], and there have been initial thoughts on embedding declarative networking into DTN [8], to the best of our knowledge, our work is the first to enable declarative routing into a DTN architecture overlayed over a MANET substrate. For the purpose of prototyping, we leverage AODV as the MANET routing protocol for propagating routing predicates to all MANET nodes.[1]

Our Contributions: We integrate two different network architectures to provide architectural support for a new capability: the content-aware DTN overlay exercising routing control over (within) the underlying (often resource constrained) MANET substrate. To this end, we make the following contributions:

- We provide a reliable DTN neighbor discovery mechanism that leverages AODV's HELLO messages to propagate DTN node names. The convergence layer of the DTN stack then maintains the mappings from DTN node names to IP (MANET) node addresses.
- In addition to DTN node names, AODV's HELLO messages are also used to propagate low-level MANET routing predicates. These latter predicates are mapped by the convergence layer from given DTN-level requirements on routing content.
- As a proof of concept, we implemented our architecture on our UML based testbed [2] that simulates a network of emulated DTN-MANET nodes as well as MANET-only nodes. The wireless connectivity and mobility of nodes are simulated using the *ns-2* simulator (ns-emulation version). The emulation uses UML (User Mode Linux) [3] to run real DTN (reference implementation [4]) and MANET (AODV routing) code.
- We present in Section V preliminary throughput results showing the efficient and correct operation of propagating routing predicates. We demonstrate two applications (case studies). The first application re-directs content to an intermediary node for pre-processing. As a side effect, our results demonstrate the performance benefit of content re-routing that dynamically (on-demand) breaks the underlying end-to-end TCP connections into shorter-length TCP connections. We also demonstrate the correct operation of a second application where a malicious node is isolated by dropping all its packets.

Paper Overview: The paper is organized as follows: in Section II we introduce the idea of predicate routing for DTN over MANET. In Section III we dissect the components of our *PreDA* architecture. Section IV describes the propagation protocol for routing predicates and Section V describes the correctness and performance testing of our approach. In Section VI we discuss related work and Section VII concludes the paper.

II. PREDICATE ROUTING IN ACTION

Quoting Roscoe *et al.* [1], predicate routing defines *"the state of the network declaratively as a set of boolean expressions associated with links which assert which kind of packet can appear where."*

For a user, a predicate is a high-level constraint on the routing of content, injected by any DTN node in the network, *e.g.*, direct all images captured by the camera on node S to DTN node I for pre-processing and authorization before sending them to the user at node D.

---

[1]Note that not all MANET nodes have to be DTN enabled but should be predicate routing enabled.

From the network point of view, the predicate is a set of rules that each MANET packet has to satisfy. Our system maps declarative user policies to such network-level routing predicates. Predicates get propagated and installed as MANET-level forwarding rules. Any DTN node can inject, from the application level, a routing rule that gets seamlessly translated into a MANET-level rule, enabling new MANET routing instructions.

We next show two examples (Tables I and II) of routing predicates.

A. Predicate Examples

Consider the injection of two predicates as in Table I. The objective of these two rules is to re-direct traffic destined to a node D to an intermediate DTN node I for authentication or pre-processing. The first rule directs all data destined to node D but not yet pre-screened at node I to node I. Notice that if the destination node D is in the path to reach the intermediate node I, then node D forwards the MANET packets matching this rule without reassembling the associated data message for pre-screening.

The second predicate ensures that pre-screened data coming from the intermediate DTN node I, reach the original destination D. Note that if nodes (S, I and D) are DTN nodes, then these predicates override the normal DTN routing process. In particular, under normal DTN routing, node I, not recognizing itself as the destination, would have directed received bundles to D without preprocessing. Thus, our *PreDA* architecture supports predicate routing at the DTN level as well.

Predicate	Action
src = $\neg I \land$ dest= D	to I
src = $I \land$ dest= D	to D

TABLE I
DIRECT ALL D-TRAFFIC TO AN INTERMEDIATE DTN NODE I FOR CONTENT AUTHENTICATION.

Predicate	Action
src = $\neg W \land$ dest= D	drop

TABLE II
DROP TRAFFIC NOT ORIGINATED BY A *White list W* OF MANET NODES AND DIRECTED TO A PRIVATE NODE.

Consider now the injection of the predicate in Table II. In this second use case, a predicate drops all the unsafe MANET traffic coming from a *black list* of IP addresses directed to a private node D. In both cases, nodes not yet aware of the injected predicate follow the normal routing behavior.

B. PreDA Advantages

Our *PreDA* architecture enables cooperation between the DTN overlay and the underlying MANET network. This co-operation includes (i) gathering information from the MANET level, (ii) applying data mining techniques at the DTN level,
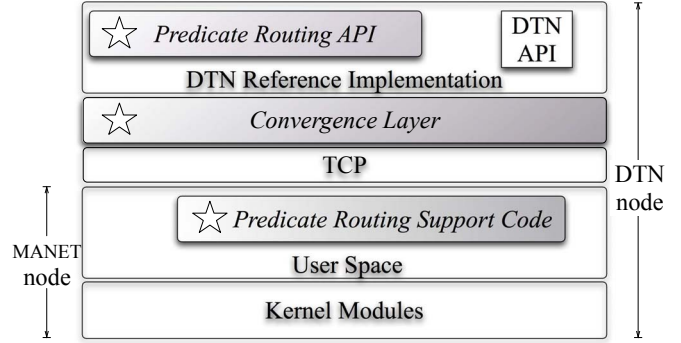

Fig. 1. Architecture of a DTN node running over a MANET substrate. The node is Predicate Routing (PR) enabled.

and (iii) declaratively generating routing rules from the DTN overlay, thus facilitating general-purpose network applications. For example, a DTN network could use trust-based approaches to collect trust information and generate routing rules as in Table II, or a sensed signal coming from a node can trigger the generation of new routing predicates (rules) for the purpose of energy saving, load balancing or data processing.

III. OVERALL ARCHITECTURE

Figure 1 shows the DTN-MANET stack—our modified and added components are marked by "stars."

The block named Application Programming Interface (API) of the DTN reference implementation is extended to allow applications to inject high-level routing requirements or constraints. We refer to this modification as *Predicate Routing API* (*PR-API*).

The *Predicate Routing Support Code* (*PRSC*) component, implemented in the routing protocol user space, mainly implements two functionalities: (1) it uses the *iptables* Linux facility [9] to install predicate MANET routing rules, so that MANET packets carrying content —IP packets or DTN bundles— are routed based on DTN-level routing constraints, and (2) it creates and manages new routing extensions to discover other DTN nodes and propagate MANET routing predicates.

The *Convergence Layer* (*CL*) interfaces DTN and MANET by maintaining the mapping between DTN node names and IP/MANET addresses. The mappings are used to translate routing predicates on DTN node names to routing predicates on corresponding IP/MANET node addresses.

A. Predicate Routing Support Code (PRSC)

Dissecting the main architecture components we have modified and integrated together, in a bottom up approach, we start by describing in detail the *PRSC*.

Our *PreDA* architecture allows every node to behave as a pure MANET node or, when needed, to use the DTN properties. This flexibility is achieved by configuring on demand the properties that the user or application wants to embed in such nodes. Listing 1 lists the options that the routing protocol deamon of *PreDA* supports for the case of the AODV protocol.

DTN EID	IP Address	Port
dtn://node1.dtn	10.0.0.1	5000
dtn://node2.dtn	10.0.0.2	5000
dtn://node7.dtn	10.0.0.7	5000
dtn://node10.dtn	10.0.0.10	5000

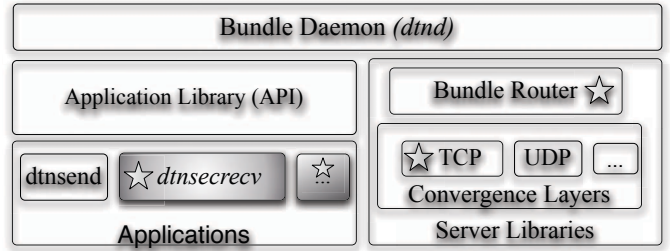TABLE III
DTN-MANET ADDRESS MAPPING.

Fig. 2. Architecture of a DTN reference implementation. The modules enabling Predicate Routing are marked with a star. Darker blocks represent applications enabled by *PreDA*.

All the nodes have to have a way to advertise two types of information: (i) DTN discovery, and (ii) predicate routing. In the particular case of AODV, this information is piggybacked as additional extensions to the HELLO message.

```
## DTN Discovery ##
-e,         --use-dtn-eid          EID
-p,         --use-dtn-port         PORT
## Predicate Routing ##
-a,         --predicate-enabled
-b,         --discovery-HELLO      N
```

Listing 1. Routing protocol deamon options.

Information for DTN discovery consists of, among other details that we omit in this discussion, (1) the DTN *endpoint identifiers (EIDs)* or addresses, specified in a URI-style format, and (2) the port that the EID will use for its communication. This information gets propagated over the network to other nodes, if and only if the node runs the DTN reference implementation (details in Section V-A).

On the other hand, information for predicate routing needs to be supported and advertized in every node. In the AODV example shown in Listing 1, routing predicates are attached into new HELLO message extensions (with option *-a*), and they get disseminated every N seconds (option *-b*).

B. Convergence Layer

The convergence layer manages connections and inter-process communications between the *DTN bundle layer* and the underlying transport protocol required by the application. *PreDA* convergence layer accomplishes the following two tasks: (1) to manage and to maintain an up-to-date DTN-MANET address mapping (*e.g.*, Table III), and (2) to convert declarative routing instructions coming from the DTN bundle layer into imperative *iptables* rules [9] (details in Section IV-B). Note that if a predicate routing rule is created to re-direct some traffic, then associated control messages, *e.g.*, keep-alive messages, would also be re-directed.

C. Predicate Routing API

In Figure 2, we zoom in on the DTN reference implementation and *Predicate Routing API* (Figure 1). Our modified and added components are marked by "stars." Blocks on the left half of the figure (*dtnsecrecv*, ⋯) represent places where *PreDA* users would add applications, such as the content re-routing and filtering applications demonstrated in this paper. Other blocks constitute our *PreDA* modification to the DTN reference implementation that users do not need to modify while embedding their applications. A fundamental DTN characteristic is that, when a node receives a bundle, it is able to store it and wait until a valid path to the destination is found or restored. When a bundle arrives at a DTN node I, which is not the final destination, the bundle enters the so-called *pending state*. The Bundle Router, handling the bundles in *pending state*, is responsible for the configuration of DTN routing. We had to modify this module to support *PreDA* applications that require predicate routing at the DTN level as well, where a data bundle received by an intermediate DTN node can be processed *before* being forwarded to its final DTN destination.

IV. PREDICATE ROUTING DESIGN

In this section we describe *what* has to be transmitted (Section IV-A), and *how* to support predicate routing in DTN over MANET (Section IV-B). Although we leverage AODV, the architecture is not tied to AODV or specific dissemination mechanism. Any mechanism for propagating predicates and DTN names could be supported, for example through link-state update messages of a link-state routing protocol (*e.g.*, using efficient multi-point flooding relays in OLSR).

A. Packet Extension

Let us consider the predicate routing packet extension in Figure 3. The extension is added to the HELLO message of the AODV protocol. The version used to build the packet extension has been the standard IPv4; the packet extension could easily be extended to IPv6 but to date *PreDA* does not support IPv6 addresses.
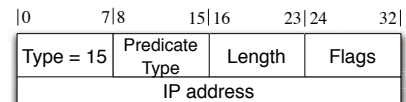

Fig. 3. Predicate routing packet extension.

The field $Type$ (chosen to be 15 just because it was one of the available IDs) specifies that the packet is a predicate routing packet. The *Predicate Type* field is used to call the right DTN/MANET application. The field $Length$ is the length in bytes of the packet extension and the last *Flags* bits together with the IP address fields, are used to build routing predicates. For example, if we want to install the predicate

described in Table I, the first rule has to be encoded as follows: The predicate type field would contain a code referring to "intermediate routing", namely, to the application "dtnsecrecv" (Figure 2). Intermediate routing predicates are of the form $S_1 \wedge D \Rightarrow I$ as in Table I, so three IP addresses are needed: one for the source, one for the authenticator and one for the destination node. Thus, the *length* field would be 32 x 4. The *flags* field in this case will be used to indicate the boolean logic of the predicate, and so we have three bits since the predicate may impose a logic "not" before each IP address. Notice that the existence or lack thereof of boolean logic negations is enough to represent every possible logic predicate on the MANET addresses that follow.

In addition to the predicate routing extension depicted in Figure 3, *PreDA* needs to propagate another extension with the HELLO message, the *DTN neighbor discovery extension*. The DTN neighbor discovery is used to disseminate the DTN *EID* of a newly discovered node, its MANET address, and the distance in hops to every other node in the network since we are using a distance vector protocol at the MANET level.

J. Ott *et al.* were the first to introduce in [10] a DTN discovery protocol. They use the discovery extension #29 in the route-request (RREQ) and route-reply (RREP) packets of the AODV protocol. Thus, DTN-capable nodes are only discovered as AODV attempts to find routes to destinations on demand. This approach may fail to discover DTN-capable nodes or may discover them in a non-timely fashion. This limitation is more serious or unacceptable when one wants to propagate routing predicates to exert control over the underlying MANET routing. In this case, timely dissemination of predicates to all MANET nodes for consistent/reliable routing is crucial. To this end, we include both AODV extensions in the AODV HELLO message, which is periodically advertised (every second by default). Whenever a node comes in contact with another, they exchange in their HELLO messages their knowledge of their mappings of DTN names to IP (MANET) addresses, as well as routing predicates.

B. Mapping DTN Constraints with Netfilter

One of the key contributions of this work is the propagation of high-level DTN constraints, in form of routing predicates, to the MANET level. To achieve this goal, the MANET network leverages *iptables*, a set of hooks provided by Netfilter [9].

To have a better understanding of how exactly predicate routing translates DTN-level rules into MANET routing decisions, let us consider the first predicate depicted in Table I. (We omit the second rule since it is similarly specified.) Let us also assume that the intermediate node has MANET address 10.0.0.1, and that the destination address is 10.0.0.2. The convergence layer maps the predicate to the following Netfilter *iptables* rule:

```
iptables −t nat −p tcp −dport 5000
−I OUTPUT −s ! 10.0.0.1 −d 10.0.0.2
 −j DNAT −−to−destination 10.0.0.1
```

Listing 2. Authenticate packets going to 10.0.0.2.

This iptables rule re-directs outgoing TCP traffic from each node, with the TCP connection established on port
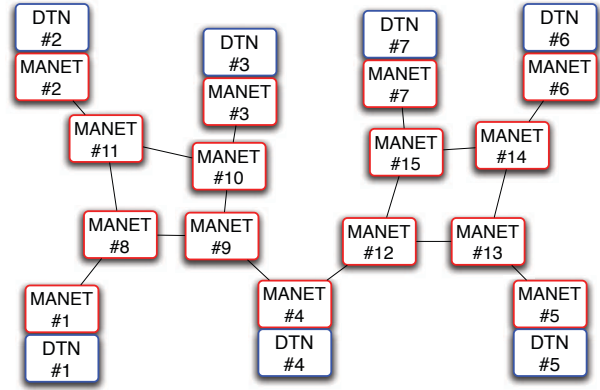

Fig. 4. *PreDA* validation scenario. AODV is the MANET routing protocol used.

5000 (the port on which the DTN process is listening), to a new destination; in particular, traffic to destination 10.0.0.2 (MANET address of DTN node #2 in Table III), to the MANET address 10.0.0.1 (corresponding to the DTN node #1).

In a further example, a DTN node, blacklisting the DTN node whose MANET address is 10.0.0.5, injects a predicate similar to the one in Table II. Therefore an *iptables* rule, as in Listing 3, is generated by the convergence layer. The *iptables* rule executes in this case a drop action (*-j DROP*) for all the packets having node 10.0.0.5 as source (*-s*). Further details (*e.g.*, why we need the option "-I INPUT 1") on *iptables* can be found at [9].

```
iptables −t raw −p tcp −dport 5000
−I INPUT 1  −s  10.0.0.5 −j DROP
```

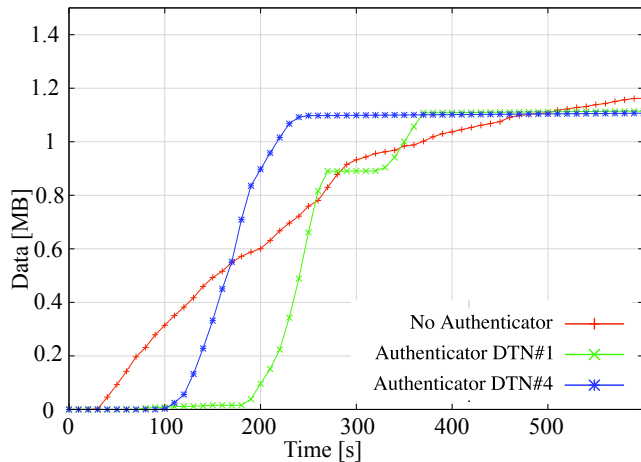Listing 3. Drop packets coming from 10.0.0.5.

It is worthy of notice that since *PreDA* has the DTN discovery extensions attached to the HELLO messages (running over UDP), all connections between neighboring nodes are kept alive even when an *iptables* rule blocks TCP traffic.
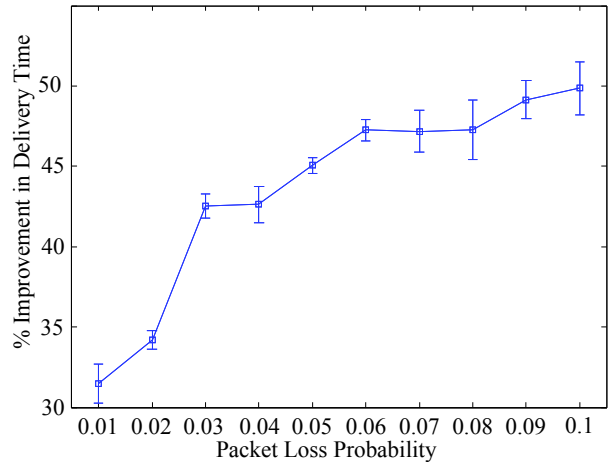
V. TESTING PreDA ON AODV

In this section we describe our validation of our *PreDA* architecture, done using our emulation testbed [2]. Starting with a description of how a routing predicate is propagated, installed and used with a reactive MANET routing protocol such as AODV, we continue the validation by injecting into the DTN-MANET network both the predicate described in Table I and, in a second scenario, the predicate described in Table II. For our experiments we have used the static topology depicted in Figure 4.

A. Predicate Propagation

Consider a management application injecting a routing predicate as in Table I, that directs all DTN bundles from DTN node *S* (node #6) to an authenticator DTN node *I* before being routed to destination DTN node *D* (node #2).

Fig. 5. (a) Data delivered at the destination vs. time for 2% packet loss probability. A predicate of the type defined in Table I is injected with different position of the authenticator DTN node. (b) Percentage improvement in data delivery time vs. packet loss probability with 95% confidence interval when a predicate of the type defined in Table I is injected.

Initially, every DTN node advertises its presence on the MANET network. This is done by attaching its DTN node name to the (periodic) HELLO message (using extension #29). The mapping between DTN name and IP (MANET) address ($e.g.$, $dtn://nodeI.dtn \iff 10.0.0.2$), is learned by neighbor nodes (and maintained by their Convergence Layer), which in turn advertise their mappings to their neighbors, and so on.

The *Predicate Routing API* creates the routing predicate after mapping the DTN names to IP (MANET) addresses by consulting the *Convergence Layer*. Then, the *PRSC* component creates the new predicate extension attaching the MANET-level routing predicate to the outgoing HELLO message. Upon receiving a MANET routing predicate, a MANET node installs that predicate using the *iptables* Linux facility.

Once the routing predicate, "src = $S \wedge$ dest = $D \rightarrow$ to I", has been propagated through the network, whenever node S sends a bundle to node D, the MANET packets carrying the bundle are first directed to node I.

The authentication application running at node I listens to DTN bundles in promiscuous mode. Once a bundle is received, it gets processed and authenticated by the authentication application. Meanwhile, the authentication application at node I installs a local DTN-specific predicate to drop the original copy of the bundle from node S—this capability is implemented by mainly modifying the *should_fwd* forwarding method in the DTN reference implementation. After authentication is done (cf. $dtnsecrecv$ in Figure 2), the bundle is then forwarded to the original destination at node D, which is realized by the installed MANET routing predicate "src = $I \wedge$ dest = $D \rightarrow$ to D". Note that a bundle sent from node S to node D through node I is reliably transmitted over two separate underlying TCP connections.

B. Throughput Improvement

Figure 5 (a) shows the data delivered at the destination for a 2% value of link loss probability when a predicate of the type defined in Table I is injected. Each point in the plot was obtained by averaging five independent runs. The plot shows a delay in receiving data at the destination when going through the two different intermediary authenticator nodes as the bundle gets reassembled from the MANET packets and gets processed. However, the 1MB-data is ultimately delivered earlier at the destination because data gets transported over two separate shorter-length TCP connections, which perform better in terms of both throughput and goodput especially over lossy wireless hops.

C. Data Delivery

Figure 5 (b) shows the percentage improvement in delivery time for varying packet loss probability when a predicate of the type defined in Table I is injected. As losses increase, the gain from breaking the source-destination TCP connection into two TCP connections through an intermediary node tends to increase. Values are computed with 95% confidence interval. Note that this performance improvement is achieved in the presence of the (small) overhead associated with supporting predicate routing.

D. Drop Unwanted Traffic

In Figure 6 we show our second validation scenario. Two DTN nodes (#4 and #6 in Figure 4) send data to DTN #1. DTN #3 injects a dropping predicate (Table II): "bundles coming from DTN #4 have to be dropped". Then a second predicate of the same kind is disseminated: "bundles coming from DTN #6 have to be dropped". Both "dropping" predicates are later removed. As we can see, during the time the predicates are installed, no data arrives at the destination.
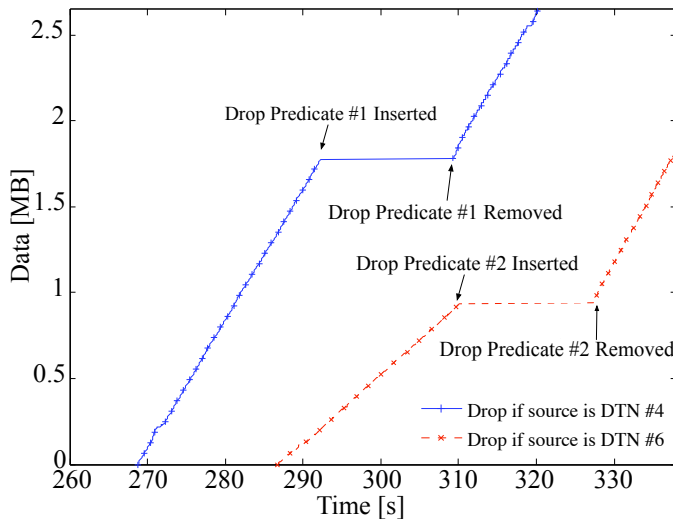
Fig. 6. Throughput drops after the "dropping" predicates propagate, and resumes after they are removed.

VI. RELATED WORK

Declarative Architectures and Protocols: A declarative networking system was developed in [11]. More recently, a clean-slate declarative transport architecture was proposed in [7], where transport rules are defined using a declarative language, Network Datalog (NDlog). The same language was recently used for declarative routing in a MANET scenario [12]. None of these approaches consider DTN.

DTN Routing: Many routing protocols for DTN have been proposed (*e.g.*, [13]). Being routing protocols for overlays, they are not designed to take into account the underlying technologies when making routing decisions, unlike our predicate routing system.

DTN/MANET: A successful attempt in integrating DTN and MANET architectures was presented in [10]. Their results have strongly inspired this work. Our architecture adds reliability properties in the DTN node discovery process as explained in Section IV, and supports a declarative routing system based on predicates. The idea of a declarative approach that defines routing constraints is not a new idea [6], but the concept of enabling a controlled network, as discussed in [1] by Roscoe *et al.*, had not been embedded in the DTN architecture before.

VII. CONCLUSIONS

In this work we designed, implemented and validated on our own testbed [2], *PreDA*, a predicate routing architecture for Delay Tolerant Network (DTN) over an underlying Mobile Ad hoc (MANET) substrate. *PreDA* allows users to declaratively express constraints on how "content" should be routed. We demonstrate two applications for content re-routing at the DTN level, and dropping traffic from malicious nodes. *PreDA* implements Predicate Routing, where high-level constraints of DTN nodes are mapped into low-level routing predicates within the MANET nodes.

In our initial prototype, we use the On Demand Distance Vector (AODV) routing protocol at the MANET level. We use the network simulator *ns-2* (ns-emulation version) to simulate wireless connectivity of both DTN and MANET nodes.

In this new setting of MANET management by the DTN overlay to improve either MANET or DTN performance, we believe that the description of the architectural components we added, the availability of our code at `http://csr.bu.edu/preda/`, and the two applications (case studies) we provided, open up possibilities for investigating many other applications. Future work includes support for such applications and optimizing our implementation.

VIII. ACKNOWLEDGMENT

REFERENCES

[1] T. Roscoe, S. Hand, R. Isaacs, R. Mortier, and P. W. Jardetzky, "Predicate Routing: Enabling Controlled Networking," *Computer Communication Review*, vol. 33, no. 1, pp. 65–70, 2003.
[2] G. Ferrari Aggradi, F. Esposito, and I. Matta, "Supporting Predicate Routing in DTN over MANET," in *CHANTS '08: Proceedings of the third ACM Workshop on Challenged Networks at MOBICOM*. San Francisco, California, USA: ACM, 2008, pp. 125–128.
[3] J. Dike, *User Mode Linux*. Prentice Hall, 2005.
[4] DTN Research Group: http://www.dtnrg.org/wiki/Code.
[5] K. 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*. New York, NY, USA: ACM, 2003, pp. 27–34.
[6] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan, "Declarative Routing: Extensible Routing with Declarative Queries," in *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 2005, pp. 289–300.
[7] K. Mattar, I. Matta, J. Day, V. Ishakian, and G. Gursun, "Declarative Transport: A Customizable Transport Service for the Future Internet," in *Proceedings of the 5th International Workshop on Networking Meets Databases (NetDB 2009), co-located with SOSP 2009*, Big Sky, MT, October 2009.
[8] E. Yoneki and J. Crowcroft, "Towards Data-Driven Declarative Networking in Delay Tolerant Networks," in *DEBS 08 International Conference on Distributed Event-Based Systems*. ACM, 2008.
[9] http://www.netfilter.org/.
[10] J. Ott, D. Kutscher, and C. Dwertmann, "Integrating DTN and MANET Routing," in *CHANTS '06: Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks*. New York, NY, USA: ACM, 2006, pp. 221–228.
[11] P2: Declarative Networking: http://p2.cs.berkeley.edu/.
[12] C. Liu, Y. Mao, M. Oprea, P. Basu, and B. T. Loo, "A Declarative Perspective on Adaptive MANET Routing," in *PRESTO '08: Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow*. New York, NY, USA: ACM, 2008, pp. 63–68.
[13] T. Matsuda and T. Takine, "(p,q)-Epidemic Routing for Sparsely Populated Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 783–793, June 2008.