Storage Aware Routing Protocol for Robust and Efficient Services in the Future Mobile Internet*

Nehal Somani, Abhishek Chanda, Samuel C. Nelson, Dipankar Raychaudhuri
WINLAB, Rutgers University

Abstract—**The prominence of wireless, mobile devices on the Internet today has motivated numerous protocols and architectures, such as the MobilityFirst Future Internet Architecture project. In this work, we present a robust, local-scale, storage-aware routing approach, called GSTAR, for use in MobilityFirst networks. GSTAR unifies techniques from MANET and DTN routing protocols. This unification with in-network storage enables it to overcome mobility-related challenges such as link quality variation, node disconnection, and network partitioning. Through NS3-based simulation, we show that GSTAR outperforms traditional link-state protocols for both wireless and hybrid wired-wireless network environments.**

I. Introduction

The recent proliferation of wireless, mobile devices necessitates flexible, efficient and robust support of mobility services in the future Internet [1]. To address the challenges associated with the future mobile Internet, several "clean-slate" proposals have recently been funded by government and private entities [2], [3], [4], [5], [6], [7]. One NSF-funded Future Internet Architecture (FIA) project, called *MobilityFirst*, is geared around the principle that mobile devices and associated applications must be treated as first-class Internet citizens. Traditionally, the challenges associated with mobility and wireless communication were partitioned from the core Internet and handled as a "last hop" problem. However, with the prevalence of hand-held and other wireless devices, we envision a future Internet where networks are not strictly characterized (e.g., core vs. access, wired vs. wireless, ad-hoc vs. managed) but are fluid and highly heterogeneous.

A few unique challenges associated with mobile devices are varying levels of disconnection, multi-homing, and dynamic formation of networks. While subsets of these challenges have been considered for ad-hoc and disruption-tolerant networks (DTNs), a unified approach within a future mobile Internet framework has not. To this end, this paper expands the GSTAR (*generalized storage-aware routing*) protocol [8] proposed in conjunction with the *MobilityFirst* architecture. In particular, we present and evaluate a unified intra-domain routing protocol capable of achieving high performance across a wide range of mobile environments, such as wireless mesh, wireless ad-hoc, DTN, and even relatively stable wired networks.

The main observation behind GSTAR is that the addition of *in-network storage* enables routers to handle network fluctuations. GSTAR operates on the principle that mobility-related challenges are best handled directly at the networking layer itself. GSTAR is essentially a *storage-aware* link-state routing protocol that enables routers to temporarily store and/or replicate data in response to detected network problems.

The main contributions of our work are three fold. First, we present the design of GSTAR, an intra-domain storage-aware routing protocol that proactively and intelligently makes path selection and transmission decisions based on network factors such as link quality fluctuation, congestion, and disconnection. Second, we present both comparative and parameter space exploration results from NS3 simulations of GSTAR. Third, we explore a robust storage-aware path selection metric that can react to fluctuations in the network.

The remainder of this paper is as follows. Section II explores the challenges and design goals for routing in the future mobile Internet. Section III presents the GSTAR protocol as well as a generalized model for path selection in storage-aware routing systems, and shows how to instantiate such a model in GSTAR. Section IV presents a comparative, NS3-based evaluation including an exploration of the parameter space of GSTAR. Finally, Section V concludes the paper.

II. Supporting Mobility in the Future Internet

The future Internet will contain end-points, and even networks, that are highly mobile, forcing a redesign of both core and edge protocol. This section presents a brief overview of *MobilityFirst*, a future Internet architecture project which provides the framework and architecture for GSTAR. Following this, challenges and design decisions of local routing in the future Internet are explored, providing motivation for GSTAR.

A. *MobilityFirst*

MobilityFirst takes the stance that applications should be able to communicate directly with *abstract entities*, including end devices, content, and context. Furthermore, the application, and even the entire end client, should not care *where* the entity currently is. Instead, the network itself should resolve physical routing identifiers for the entity and progress the message. This architecture brings together five core principles: (1) separation of naming and addressing, (2) support for late or progressive binding, (3) support for group-based paradigms, such as anycast and multicast, (4) the ability for the network itself to deal with all forms of mobility in a unified manner and (5) hop-by-hop transport of large data units.

In *MobilityFirst*, all entities are named via a flat, location independent structure called the *globally unique identifier*, or GUID. A large-scale, distributed system called the *global*

*Research supported by NSF Future Internet Architecture (FIA) grant CNS-1040735

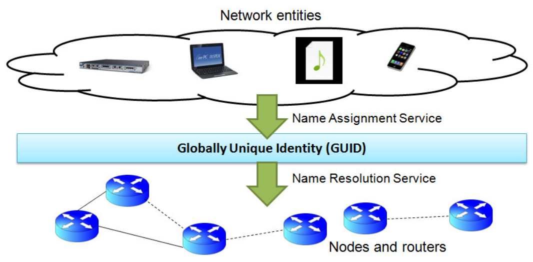


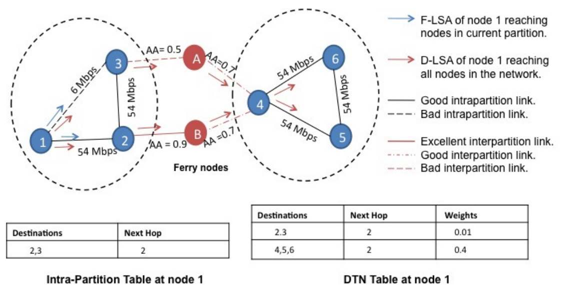
Fig. 1. MobilityFirst Architecture

name resolution service, or GNRS, is responsible for binding GUIDs to routable addresses and is directly queryable by routers. This is illustrated in Figure 1. Routing protocols take advantage of in-network storage and the ability to dynamically query the GNRS to update the GUID-to-address mappings when needed. For more information, refer [9], [8].

B. Challenges in Local-Scale Routing

Mobility of wireless devices leads to a number of challenges not addressed by current local-scale (intra-domain, ad-hoc, mesh, etc.) routing protocols, such as a high degree of bit rate fluctuation, multi-homing, and node or network disconnection. The DTN community has used techniques such as message replication [10], [11], [12] and hop-by-hop transport [13] to help address some of these challenges and bridge partitions in the network. To a lesser degree, the ad-hoc community has explored storage-aware routing in a link-state fashion, particularly with CNF [14]. GSTAR is a unified solution bridging the two domains and the usage scenarios in-between. While merging DTN and MANET protocols has been considered in literature, these approaches usually require specialized nodes (e.g., ones with DTN capability) [15], [16], or are simply using one approach to extend the other [17]. By unifying techniques from MANET and DTN protocols, GSTAR is able to smoothly transition between both types of environments. Further, GSTAR has been designed to work well in access networks with a mix of wired and wireless components, and functions as an efficient intra-domain routing protocol similar to OSPF when the network is mostly well-connected.

The key to supporting unpredictable mobility in the future Internet is to handle the challenges of mobility *directly at the network layer*. In order to give routers the freedom to handle mobility challenges directly, path quality information must be exposed to the routing protocol, and resources must be available to respond to problems. In particular, *storage* is a highly useful resource when handling the challenges of mobility, as shown by the DTN community, and is relatively cheap. We envision future routers to have an abundance of storage directly accessible by the network layer. GSTAR aims to directly address the challenges of mobility from both an ad-hoc and DTN perspective by providing both fine-grained topology and link quality information to nodes within a single partition, as well as course-grained connection probability information to all nodes in the network.


F-LSA of node 1 reaching nodes in current partition.
D-LSA of node 1 reaching all nodes in the network.
Good intrapartition link.
Bad intrapartition link.
Excellent interpartition link.
Good interpartition link.
Bad interpartition link.
1
2
3
4
5
6
A
B
6 Mbps
54 Mbps
54 Mbps
54 Mbps
54 Mbps
54 Mbps
AA= 0.5
AA=0.7
AA = 0.9
AA =0.7
Ferry nodes

Destinations	Next Hop
2,3	2

Intra-Partition Table at node 1

Destinations	Next Hop	Weights
2.3	2	0.01
4,5,6	2	0.4

DTN Table at node 1

Fig. 2. GSTAR

III. Robust Routing with GSTAR

GSTAR is a combination of techniques used in MANET and DTN routing protocols. This merging of protocols with in-network storage enables it to overcome mobility related challenges such as link quality fluctuations, node disconnection, and network partitioning. GSTAR provides next-hop information based on the current link qualities between the nodes in the network. The routing decisions are made on set of data packets called *chunks*. These chunks are transmitted reliably in a hop-by-hop fashion under a GUID-based destination. In this section, we present GSTAR details.

A. Protocol Overview

Each node in the network uses two types of topology and path quality information (Figure 2) while deciding to proactively store or push out messages. The first, called intra-partition graph, is maintained via *Flooded-Link State Advertisements (F-LSA)*. F-LSAs carry fine-grained, time-sensitive information about the links in the network. The second, called DTN graph, is maintained via epidemically *Disseminated-Link State Advertisements (D-LSA)*. D-LSAs carry connection probabilities between all nodes in the network.

The autonomous unit of data transmission at the network layer is the *chunk*. Each node first searches its intra-partition table for the destination address and proactively stores the chunk if the path is of low quality. If no end-to-end path exists for the destination; the node checks the DTN table. In this case, the node proactively pushes the message by using a probabilistic view of the network.

B. Proactive Control Message Dissemination

All GSTAR nodes periodically broadcast link probe messages to learn about their current 1-hop neighbors and corresponding link quality estimates in the form of Expected Transmission Time (ETT). This enables a node to update a running database of contact probabilities with all other nodes.

All nodes periodically flood F-LSA messages containing *time-sensitive, fine-grained* information regarding the local 1-hop neighbors. These messages are received only by the nodes in the current partition. For example, in Figure 2 F-LSA of node 1 are received by node 2 and node 3 only. Each F-LSA message contains the node's current storage availability, a list of current 1-hop neighbors, and the short and long term ETTs associated with each of these 1-hop neighbors. The short term ETTs can be computed locally by taking the average of the last three ETTs. Long term ETTs are computed as an average of short term ETTs. Section IV provides a simulation-based exploration of the performance gains related to how LETTs are computed. As as baseline, we assume $LETT = \alpha \cdot SETT + (1 - \alpha) \cdot LETT$, where α is a weighting factor we explore via simulation. Intuitively, capturing the quality of a link over time would depend on many factors such as if the link is *periodically* bad or not. If so, then more weight should be placed on past values. However, this is not true if the link abruptly becomes poor, since past information is less relevant.

Finally, in order to bridge partitions that may form in the network, all nodes epidemically disseminate information about connection probabilities between all other nodes in the network. When a node receives a D-LSA message, it not only sends it to all of its neighbors, but also carries it indefinitely, giving a copy to all other nodes it meets. Thus, in the network of Figure 2, D-LSA of node 1 is received by all nodes in the network. The connection probability computation is based on the idea of *average availability*, presented in PREP [12]. The basic idea is for all nodes to keep track of the percentage of time connected to every other node in the network.

C. Data Transmission

SETT values in F-LSA messages are used to compute the *intra-partition forwarding table*. Thus node 1 chooses a 2-hop path of node 2 - node 3 instead of the 1-hop direct path. This path selection metric is sufficient for most wired/wireless hybrid networks; however, for larger, ad-hoc networks a more intelligent storage-aware approach is needed. We defer this discussion until the next subsection. The *DTN forwarding table* is computed by using a function of the average availability (AA) in D-LSA message as weights. The approach taken by PREP [12] is to define a weight as $1 - AA + 0.01$, where the small constant is added to favor paths with shorter hop counts if the AAs are all close to 1. The next-hops are computed using these weights as metrics for a shortest path algorithm. Thus, in Figure 2, the *intra-partition forwarding table* at node 1 contains paths for nodes 2 and 3. The *DTN forwarding table* of node 1 consists of paths for all nodes in the network.

If an intra-partition path exists to the destination, the node checks the SETT and LETT values along the path to decide if the current path quality is abnormally bad or not. The *short term path quality* (STPath) and *long term path quality* (LTPath) is the sum of SETT and LETT values along the path, respectively. A decision to *store* a chunk is made locally at every hop if $STPath > k \cdot LTPath$, else, a *forward* decision is made. Note that k is a threshold detecting how bad the link must be, relative to its long term performance, before a store decision is made; 1.1 is used as a baseline, although we are investigating a dynamic approach to selecting k. Mostly, the STPath and LTPath look similar to all hops in the route. Thus, once nodes close to the source (or the source itself) see the path as improved, others along the route will not store. This proactive storing of data close to the source helps alleviate congestion, minimize low-level transmissions, and saves storage space in congested areas. This approach was used in CNF routing [14]. If the destination is not found in the intra-partition forwarding table, then the DTN forwarding table is consulted. In this case, if the next hop is available, the chunk is immediately forwarded to it.

D. Extension: Storage-Aware Path Selection

In large, ad-hoc networks with multiple paths with fast changing qualities and storage availabilities, a more intelligent, storage-aware path selection technique is needed. The goal is to choose the path that minimizes the end-to-end delay, accounting for time in storage, between two nodes *within the same partition*. As a block traverses a path, each node along the path will perform the following steps, (1) store the block until ready to send, (2) attempt to gain access to the channel, and (3) transmit the block to the next node along the path.

Assume nodes along an arbitrary path are labeled $1, 2, ..., k$ where 1 is the source and k is the destination. Let $BACK_i$ be the expected amount time a block will spend in storage at node i due to back pressure from the hop-by-hop transport. Let $HOLD_i$ be the expected amount time a block will spend in storage at node i due to a routing decision to store Let P_i be the probability that a routing decision to store (not due to back pressure) is made by node i. Let $CHAN_i$ be the expected amount of time a block will wait while node i gains access to the channel to send the block. Let T_i be the expected amount of time a block will be in transit from node i to node $i + 1$. The total end-to-end latency for a block over a path is:

$$delay_{1,2,...,k} = \sum_{i=1}^{k} (P_i \cdot HOLD_i + BACK_i + CHAN_i + T_i)$$

Storage time due to back pressure at node i will directly depend on the amount of data at node i with higher priority than the block in question in addition to how long it takes for space to open at node $i+1$. Making the simplifying assumption that blocks are served on a first-come-first-serve basis:

$$BACK_i = \frac{D_i}{tx_{i,i+1}} + TWS_{i+1}(D_i)$$

where D_i is the amount of data being held at node i due to back pressure, $tx_{i,i+1}$ is the transmission rate between node i and node $i + 1$, and $TWS_{i+1}(D_i)$ is the time node i waits on node $i + 1$ to clear out D_i space. Note that the transmission time, T_i, is $T_i = \frac{blockSize}{tx_{i,i+1}}$.

We now describe how to instantiate this model with GSTAR, which takes the approach of holding data close to the source until the path quality is not abnormally bad. The following simplifying assumptions to the model are for a particular path p: blocks are held at the source, and after they are released they will not be held (except for queuing due to back pressure) at any intermediate node. Therefore, $P_i = 0$ for $2 \leq i \leq k$. $TWS_i = 0$ for $1 \leq i \leq k$, which is a result of holding the block at the source until the path is of normal quality.

The current values of D_i and $tx_{i,i+1}$ can be estimated by node i and transmitted to all other nodes via F-LSAs. The source node can estimate P_1 in the following way. $P_1 = 0$ if the current SETT path metric is normal or low relative to the LETT path metric. $P_1 = 1$ if the current SETT path metric is high relative to the LETT path metric. Furthermore, $HOLD_1$ is the average amount of time for that path that the SETT is relatively high. This can be estimated by proactively keeping track of the following: given the SETT is high, on average how long does it take to become normal or low.

With this information, a source node can compute the delay metric for any given path as follows:

$$delay_{1,2,...,k} = P_1 \cdot HOLD_1 + \sum_{i=1}^{k} (BACK_i + T_i)$$

where D_i and $tx_{i,i+1}$ are estimated from received LSA message and $TWS_i = 0 \ \forall i$.

IV. Evaluation

The primary goal of our evaluation is to show that by intelligently utilizing in-network storage, GSTAR outperforms traditional and storage-augmented link-state protocols in both wired and wireless network environments. We consider three categories of simulation: (1) connected networks with fluctuating link quality, (2) node disconnection and network partitioning, and (3) parameter space evaluation. The NS3 simulator is used for the evaluation.

A. Simulation Set-up

The first set of simulations demonstrates the effectiveness of proactively storing when the path to the destination is abnormally bad. GSTAR is compared against traditional link-state with hop-by-hop transport in both pure wireless and wired-wireless hybrid environments (see Figure 3 for the topology). For the hybrid case, only the links connecting the destination nodes (nodes 6 and 7) are wireless. To simulate congestion in the network, we vary the link quality for one of the flows by fluctuating the bit-rate of the link between node 6 and dest 1 between 54 Mbps and 6 Mbps periodically. Other links are set to 54 Mbps. Simulation run time is 90 seconds.

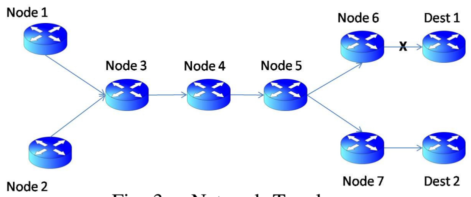


Fig. 3. Network Topology

The second set of simulations demonstrates the effectiveness of using storage to proactively push data towards the destination, even if the destination is periodically disconnected from the network. In this set, GSTAR is compared to a version of GSTAR without DTN capability (which amount to storage-augmented link-state). The same hybrid topology is used, with nodes 6 and 7 periodically disconnecting every $[15 + rand(-5, 5)]$ seconds. We also explore a network partition scenario, where two clusters of 4 nodes each (not shown due to space constraints) are bridged by a mobile ferry node. The bit rate for the links is set to 54 Mbps.

The last set of simulations explores the parameter space of GSTAR, namely computing LETT. Three cases are considered. First, LETT is an exponentially weighted moving average (EWMA) with α, the weight of the current measurement, set to 0.1. Second, LETT is an EWMA with α set to 0.5. Third, LETT is a simple average of the past 10 measured ETTs. We use the hybrid topology for this simulation.

Aggregate goodput, measured as the total number of chunks received by all destinations, is the primary performance metric throughout our evaluation. Each data point is the average of 10 runs, with 95% confidence intervals shown around it.

B. Performance Results

GSTAR is first compared to traditional link-state in both wireless and hybrid environments. For the wireless network, we have two 5-hop flows (Node 1 - Dest 1 and Node 2 - Dest 2) continuously transmitting chunks of 25 data packets. We vary the bit-rate fluctuation time from $10 + rand(-2.5, 2.5)$ seconds to $20 + rand(-5, 5)$ seconds in increments of 2.5 seconds. For the hybrid network, we consider two cases; one with 4 flows and the other with 6 flows each with chunk size of 10 data packets. In both the scenarios, the bit-rate fluctuates from 54 Mbps to 6 Mbps every $15 + rand(-5, 5)$ seconds. We vary the offered load by increasing the number of chunks generated per second, starting at 25 chunks per second by each source, followed by 50 to 200 in increments of 50.

For medium to high offered loads, GSTAR outperforms traditional link-state, as shown in Figures 5(a,b,c). The cross-over points in the two graphs signifies the load after which the network is fully utilized. After this point, the two flows starts competing with each other over network resources. With traditional link state protocol, both the flows are given equal weights. Thus, with 6 Mbps bit rate, flows for destination 1 would keep assess of the channel for a longer duration. During this period, the other flows for destination 2 cannot use the channel; even though it can send at a bit rate of 54 Mbps. Hence, network resources are not fully utilized with a traditional link state protocol and congestion in one flow affects other flows. GSTAR is able to efficiently detect this variation in link quality using its parameter set of SETT and LETT and proactively stores data. This enables it to alleviate the effect of congestion on the other parts of the network. Thus, flows for destination 2 are not affected by the bad links for flows for destination 1 resulting in considerable gain in aggregate goodput.

Next, GSTAR is compared to storage-augmented link-state (or, GSTAR without DTN capability). GSTAR with DTN proactively pushes data towards a disconnected destination, allowing it to receive data as soon as it reconnects. Without DTN, the disconnected node has to wait for its LSAs to propagate through the network before it can receive data. As shown in figure 5(a), there is evidence for a gain in aggregate goodput across all network loads. For the network partition case, where a single ferry node bridges two clusters, 3 flows are randomly sourced and sent to random destinations. These results, shown in figure 5(b), indicate that GSTAR is able to discover the ferry node and deliver messages across partitions.

Finally, a parameter exploration of GSTAR is performed, namely the computation of LETT. Two 5-hop flows are

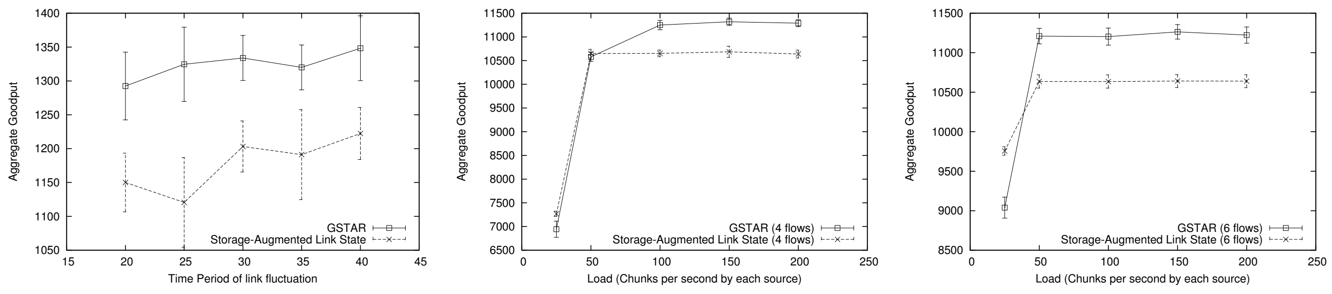


Fig. 4. GSTAR vs Traditional Link State with (a) wireless - 2 flows, (b) hybrid - 4 flows, (c) hybrid - 6 flows

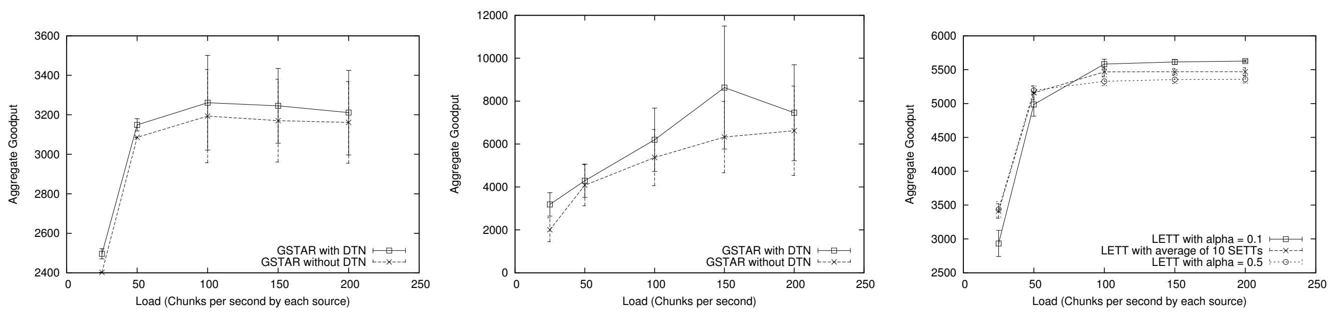


Fig. 5. (a) Node Disconnection, (b) Network Partition, (c) Computation of LETT

competing and the chunk size is set to 25 packets. As the link quality variation is periodic, the goodput is maximized by giving more weights on the past values of ETT, as shown in figure 5(c). Using the average of past 10 ETTs as LETT gives equal weights to only last 10 seconds of link qualities. This gives better results than with the moving average weight of 0.5 here because the periodicity in link quality variation is around 15 seconds. Hence, with periodicity in link quality variation, past statistics provide useful information on the general link quality. Giving more weight to past values of SETTs results in better goodput for this network.

Thus, GSTAR is able to detect different types of mobility challenges, such as link quality fluctuation, node disconnection, network partitioning, and respond to them. By utilizing storage for proactive holding or pushing, GSTAR achieves significant goodput gains over traditional link-state protocols.

V. Conclusions and Future Directions

In this paper, we have proposed and evaluated a generalized storage aware local routing protocol for the future Internet. We have shown how GSTAR can achieve significant performance gains by combining local link state and DTN style routing. Although omitted from this paper due to space constraints, we currently have a proof-of-concept implementation of GSTAR running on both the GENI [4] and ORBIT [18] testbeds. In the future, we plan to perform a comprehensive evaluation of our GSTAR implementation. We also plan to make GSTAR more robust to dynamic network conditions by varying its parameters automatically. For example, the threshold for the decision to store or forward should be dynamic. Finally, we plan to enhance GSTAR by giving it replication capabilities.

References

[1] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. *ITU T Kaleidoscope: Innov. in NGN*, 2008.

[2] NSF FIA project. http://www.nets-fia.net.

[3] Networking technology and systems: Future internet design (FIND), NSF program solicitation, 2007.

[4] Global environment for networking innovations (GENI), NSF program solicitation. http://www.geni.net, 2006.

[5] FP7 information and communication technologies: Pervasive and trusted network and service infrastructures, european commission, 2007.

[6] M. Lemke. Position statement: FIRE, NSF/OECD workshop on social and economic factors shaping the future of the internet, January 2007.

[7] New generation networks. http://www2.nict.go.jp/w/w100/index-e.

[8] Samuel C. Nelson, Gautam Bhanage, and Dipankar Raychaudhuri. GSTAR: Generalized storage-aware routing for mobilityfirst in the future mobile internet. In *Proceedings of MobiArch*, 2011.

[9] MobilityFirst. http://mobilityfirst.winlab.rutgers.edu/.

[10] J. Burgess, Br. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[11] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. ACM SIGCOMM*, 2007.

[12] Ram Ramanathan, Richard Hansen, Prithwish Basu, Regina Rosales-Hain, and Rajesh Krishnan. Prioritized epidemic routing for opportunistic networks. In *MobiOpp 07*, 2007.

[13] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: a new paradigm for wireless transport. In *Proc. of NSDI*, 2009.

[14] S. Gopinath, S. Jain, S. Makharia, and D. Raychaudhuri. An experimental study of the cache-and-forward network architecture in multi-hop wireless scenarios. In *Proc. of LANMAN*, 2010.

[15] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann. Integrating DTN and MANET routing. In *Proc. of ACM CHANTS*, 2006.

[16] G. F. Aggradi, F. Esposito, and I. Matta. Supporting predicate routing in dtn over manet. In *Proc. of ACM CHANTS*, 2008.

[17] J. Whitbeck and V. Conan. HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks. *Comp. Commun.*, 33, August 2010.

[18] Orbit. http://www.orbit-lab.org, 2011.