PhotoNet: A Similarity-aware Picture Delivery Service for Situation Awareness

Md Yusuf Sarwar Uddin, Hongyan Wang, Fatemeh Saremi, Guo-Jun Qi, Tarek Abdelzaher and Thomas Huang
University of Illinois at Urbana-Champaign
{mduddin2, wang44, saremi1, qi4, zaher, huang}@illinois.edu

***Abstract*—We propose PhotoNet, a picture delivery service for camera sensor networks. PhotoNet is motivated by the needs of disaster-response applications, where a group of survivors and first responders may survey damage and send images to a rescue center in the absence of a functional communication infrastructure. The protocol runs on mobile devices, handling opportunistic forwarding (when they come in contact) and in-network storage. It assigns priorities to images for forwarding and replacement depending on the degree of similarity (or dissimilarity) among them, such that scarce resources are assigned to delivery of most "deserving" content first. Prioritization aims at reducing semantic redundancy such as that between pictures of the same scene at the same location taken from slightly different angles. This is in contrast to redundancy among identical objects and among time series data. PhotoNet delivers more diverse pictures in terms of event coverage suppressing logically redundant content belonging to the same event. We show that, in resource constrained networks, reducing semantic redundancy can significantly improve the utility of the service.**

I. Introduction

In this paper, we develop a picture delivery service for camera sensor networks, called *PhotoNet*. The service is geared for disaster recovery scenarios where survivors and first responders survey post-disaster damage and send pictures of it from their mobile devices (such as camera phones) to a command or rescue center. We assume that infrastructure (such as power and cell towers) is down, making communication possible only opportunistically between nearby wireless battery-operated devices. As nodes move and meet other nodes, data spreads, leading to a disruption-tolerant network (DTN) model. We are interested in scenarios where bandwidth, contact opportunities between nodes, or node storage is limited, leading to resource bottlenecks that prevent delivery of all pictures to the destination. The service aims at maximizing awareness of locations that need attention. We call this metric *event coverage*. Hence, delivery of pictures from many different locations is preferred to delivery of many pictures from the same location. Similarly, delivery of dissimilar pictures from a given location (likely covering different events) is prefered to delivery of very similar pictures from that location. In the paper, we show that by prioritizing image storage and transmission depending on possible overlap with other images, one can make significantly better use of scarce resources thereby substantially improving overall event coverage at the sink. Overlap is estimated by comparing the visual content of images as well as their locations and timestamps. The service is made possible thanks to the proliferation of mobile devices with digital cameras, which makes deployment feasible.

The main contribution of PhotoNet lies in its message prioritization scheme, called CAP (Content-Aware Prioritization) that aims to maximize delivered content diversity. By maximizing diversity, the network has a better chance at giving the sink the "big picture" quicker, as opposed to delivering lots of pictorial coverage of more populated locales and none on more isolated ones.

PhotoNet's content-based prioritization scheme, CAP, bears an interesting difference from traditional traffic prioritization schemes (e.g., those discussed in network QoS literature [1], [2]). While prior schemes associate priority with each message *independently* (e.g., based on its content type, class, source, or destination), CAP considers the *relations* between different objects in assigning priority to each. Specifically, it tries to maximize a measure of distance between reported objects, hence favoring dissimilar content.

Understanding relations between objects is a necessity, not a choice, whenever network utility is *not additive* in utility derived from delivery of individual objects from different sources. For example, delivering the first picture of a damage scene may have high utility. Delivering more pictures of the same scene from other sources has progressively lower utility, since the information becomes partially redundant. This utility saturation effect calls for content prioritization schemes in which priority is not a value defined inherently for each object in isolation, but rather is a function of relations between objects (such as whether a new picture is similar to, and from the same location as, a previously delivered one).

The above observation leads to an important concern. If priority cannot generally be determined for each object in isolation, it becomes expensive to assign priorities. For example, if pairwise combinations of N objects need to be considered, the worst-case overhead is $O(N^2)$. Fortunately, in DTNs, a node spends most of its time between encounters (e.g., spends minutes or tens of minutes). Hence, the bulk of similarity processing can be done in between encounters, leading to efficient prioritization of forwarding when nodes meet. Indeed, much of the design of PhotoNet intends to minimize prioritization overhead. Our evaluation shows that despite the overhead, network performance is improved.

We simulate PhotoNet using the ONE (Opportunistic Network Environment) simulator and show that PhotoNet achieves higher event coverage compared to traditional content-agnostic protocols in the presence of scarce resources, such as poor

bandwidth, low contact durations, or limited storage. We also implemented the service on mobile phones to verify its feasibility and overheads in practice.

The rest of the paper is organized as follows. Section II describes the content storage and query mechanisms in PhotoNet. Section III describes the proritization scheme. Simulation results and performance evaluation are presented in Section IV. Section V reviews existing literature on content-based networking, data fusion problems and data forwarding techniques in DTNs. Section VI concludes the paper and presents future research directions.

II. Design of PhotoNet

In DTNs, content is usually replicated on multiple devices creating a distributed in-network storage system. Each node may carry pictures generated by the node itself or obtained from others via a replication process. Each stored picture is accompanied by meta-data that enables computing semantic redundancy. In PhotoNet, pictures are delivered against a query, issued by an authorized entity such as the command station. These queries are long-lived and stored in a query table. The default query is to obtain all pictures from all sources. In the rest of this paper, we focus on the default query. To prioritize message transmission and storage, each node also runs CAP, the content-aware prioritization scheme. It generates the order in which pictures are replicated to other nodes and the order in which they are dropped from the storage when capacity is exceeded. The components of PhotoNet are shown in Figure 1. In this section, we describe how pictures are stored, named and represented for computing semantic redundancy among them. In the following section, we detail our prioritization scheme.

A. Picture Organization and Naming

PhotoNet organizes pictures in a way that facilitates prioritization. A very general way of doing so is inspired by the content-centric networking paradigm, recently articulated by Van Jacobson [3]. In this paradigm, networks name content chunks, not machines, and queries express interest in content collections by name.

Following conventions of content-centric networking, all pictures managed by PhotoNet fall into a global naming structure that looks like a UNIX directory tree. Pictures taken by source nodes have names that place them in one of the "directories". For example, a rescue worker might take a picture, called '/rescue/pictures/volunteerA/pic1.jpg'. A fully quantified name refers to a unique item. Names can also be *partially* quantified to designate a collection of items that have the named prefix. Pictures have unique IDs, computed as a combination of device-dependent identifier (for example, IMEI of mobile devices, physical MAC address, or a long hash of the picture itself or a random string) and the local timestamp when the picture is generated. These IDs are used in detecting duplicate pictures in local storage, not as a part of their names.

Queries (interests) are expressed in terms of content identifiers or prefixes, such as '/rescue/pictures', that define a subtree in the global naming structure. The collection of pictures that belongs in that subtree is said to match the query. This collection is denoted by $pics(q)$ for a query q. Each query is associated with a sink node, to which pictures of the query are due.

In PhotoNet, queries are long-lived. They can even be issued prior to mission launch at initialization (for example, before volunteers are deployed in the field). Queries flood the network. Source and intermediate nodes determine whether a particular picture in their buffer belongs to a particular query based on whether the queried name is a prefix of the picture's name. Pictures matching each query are logically grouped by two types of linked lists, sorted by priority. One list is sorted by forwarding priority and the other list is by dropping priority. The objective from both priority orders is to reduce semantic redundancy based on content similarity. The priority order for forwarding and dropping are different because dropping priority is a function of local content only, whereas forwarding priority to an encountered node is a function of content on both nodes and needs to be computed on the fly when a node is at contact with another node. We describe the details of content prioritization in subsequent sections. Figure 1 describes, at a conceptual level, data structures used by PhotoNet.

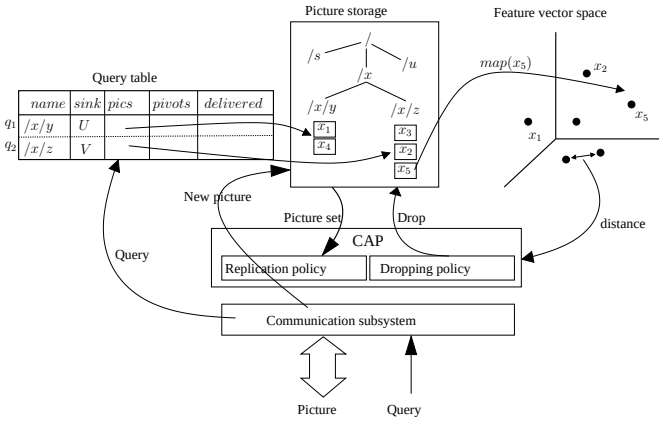


Fig. 1. PhotoNet architecture.

B. Vector Representation of Pictures

PhotoNet extracts features of pictures and expresses them as multi-dimensional vectors that we call *feature vectors*. We define a function, $map(x)$, that maps a picture, x, into a fixed-length vector in the feature vector space. Once mapped, the semantic similarity between pictures is simply reflected in *distance* between the associated points in space. Items that have similar content and taken at nearby locations are closer in the feature vector space, whereas dissimilar objects or objects from different locations lie farther apart. Any suitable distance function, say Euclidean, can be used to measure distance or degrees of similarity among items.

The components of a feature vector associated with a picture can be attributed to a set of physical properties of the picture or some raw features extracted from the image data, such as color histograms. Since we are interested in information on geographic locales, meta-data such as location and time associated with data items also serves as features in the feature

vector. Those features are very important in determining semantic redundancy. For example, different buildings may look alike in pictures. However, if their locations are different, then there is no semantic redundancy because these pictures carry information on different events. In addition, human assessment can also serve as useful input for the vector space. For example, the photographer can label his images with tags or keywords for organization purposes. In the current implementation human tags are not used.

To this end, we define $\overline{x} = \{x_1, x_2, \ldots, x_k\}$ to be the feature vector of picture x. We use both spatio-temporal attributes and image-features. Thus, the vector can be expressed as: $\overline{x} = \{t(x) \mid \bar{l}(x) \mid \overline{f}(x)\}$, where $t(x)$ and $\bar{l}(x)$ denote time and location of the picture (when and where originated) respectively and $\overline{f}(x)$ denotes the vector of visual image-features. In the current implementation, we simply use the color histogram.

C. Semantic Distance among Pictures

We define *semantic distance* to be the level of dissimilarity between pictures: larger distance means highly dissimilar and shorter distance means fairly similar. In our application context, pictures are taken of physical events. Hence, we say that pictures are similar if they can be associated with the same physical event. Accordingly, pictures generated in two relatively remote locations (say, 1km apart) or at two largely different times (say, 6 hours apart) are "dissimilar" (high spatio-temporal distance). When pictures are closer in spatio-temporal space, their similarity is further decided by the distance in the image-feature space.

In our implementation of distance between pictures, we first use the time gap between two pictures as a binary decider. If it is beyond some threshold (say, 6 hours), we assume these two pictures belong to different events and assign a very large distance value. Otherwise, the distance is given by location and image-features. Semantic distance due to location is simply Euclidean distance between location coordinates. For image-features, we compute the distribution of colors in a picture, commonly known as color histogram. For a certain color (or a color bin), i, of picture x, let $f_i(x)$ be the fraction of pixels in the picture that has color i or any color in that bin. Obviously, $\sum_i f_i(x) = 1$. We compute KL-divergence (Kullback-Leibler divergence) distance between the corresponding histograms for two pictures x and y as follows:

$$\|\overline{f}(x) - \overline{f}(y)\| = \left[\sum_i^h f_i(x) \log \frac{f_i(x)}{f_i(y)}, \sum f_i(y) \log \frac{f_i(y)}{f_i(x)}\right]$$

where $[u, v] = \max(u, v)$. By proper weighting, as shown in the evaluation (Figure 4), we can ensure that the numerical value of image-feature distance is very small compared to location distances. Therefore, location predominantly defines the semantic distance between pictures. When pictures originate near one another, the image-features further refine the semantic distance. Therefore, we use the following distance function:

$$\|\overline{x} - \overline{y}\|^2 = \alpha\|\bar{l}(x) - \bar{l}(y)\|^2 + (1-\alpha)\|\overline{f}(x) - \overline{f}(y)\|^2 \quad (1)$$

where α is a scaling factor and is given by: $\alpha = \exp\left(-\frac{\tau^2}{\|\bar{l}(x)-\bar{l}(y)\|^2}\right)$. It indicates that image-features are ignored above a certain location threshold, but become increasingly more important below that threshold when the location distance itself is ignored. The parameter τ defines this threshold and we use $\tau = 100$ meters. In the following, we use the same symbol to represent a picture and its feature vector, and use $\|x - y\|$ to denote the corresponding distance.

Although we build the service explicitly for pictures, the same architecture can be adopted for delivery of other content types, such as text, audio and video, provided that an appropriate map function and an associated distance function are defined. Since we adopt content-centric networking abstractions, content format should be implicit in its name. For example, '/rescue/audio/' could refer to audio data. A different $map()$ function should be associated with every "directory" in the content tree that knows how to map content in that directory into a space where semantic distance represents (dis)similarity. Extension of PhotoNet to other content types, however, is outside the scope of this paper.

III. The Picture Prioritization Scheme

In this section, we describe CAP (content-aware prioritization), the picture prioritization scheme used by PhotoNet. We describe the diversity measure of a picture collection and show how pictures are prioritized for storage and forwarding to achieve greater diversity at the final collection point.

A. The Diversity Measure of a Picture Set

The goal of PhotoNet is to maximize event coverage. It does so by maximizing the diversity of delivered content. This maximization requires a measure of diversity. Hence, given a set I of n pictures, its diversity, denoted as $\Psi(I)$, is computed as the average of squares of pairwise distances between all pictures.

$$\Psi(I) \quad = \quad \frac{\sum_{x,y \in I} \|x - y\|^2}{n \times (n-1)} \quad (2)$$

Note that pictures that are farther from each other produce a larger $\Psi(I)$ because they improve diversity, whereas data items that are clustered together produce a lower $\Psi(I)$ because they cover a smaller region of the data space. To reduce semantic redundancy, the network should always make forwarding and dropping decisions that generate a higher $\Psi(I)$ for a collection of pictures, I.

It is also useful to measure the contribution of a given individual picture to the diversity measure of the collection. This quantity, denoted by $\psi(x)$, is simply given by the average of square of distances with other pictures in the set, i.e., $\psi(x) = \frac{1}{n-1}\sum_{y \neq x} \|x - y\|^2$. Obviously, $\Psi(I) = \frac{1}{n}\sum_x \psi(x)$.

B. A Prioritized Dropping Policy

In cases where storage space becomes a bottleneck, a node may need to decide which picture to drop. A node never

deletes pictures that the node itself produces (for which the node is the source). We assume that nodes have enough space to hold their own pictures, but the storage for pictures that are replicated from other nodes is limited. Hence, some of these pictures may be dropped due to storage constraints. The question is to which picture to drop when storage capacity is exceeded.

The notion of the diversity $\Psi(I)$ of picture set, I, offers an answer. Namely, pictures should be dropped in an order that maximizes the diversity of the remaining set. Since different pictures have different length, it is best to normalize diversity by storage requirements. For example, removal of a long picture is preferred to removal of a short picture, if diversity of the remaining set is the same. Hence, we drop the picture that maximizes the diversity of the remaining set per byte stored. More formally, let the set of pictures stored locally at node X, that match query q, be called $pics_X(q)$. Let the total space needed to store $pics_X(q)$ be S_q^{total} and let the size of picture x be denoted by $s(x)$. For each query, q, the next picture to drop, x, among those locally stored pictures that match the query, $pics_X(q)$, is computed as follows:

$$x = \arg_{x \in pics_X(q)} \max \left(\frac{\Psi(pics_X(q) - \{x\})}{S_q^{total} - s(x)} \right) \qquad (3)$$

In the presence of multiple queries, CAP first drops pictures that do not match any query. If no such picture exists, it chooses one query at a time and drops the picture computed from Equation (3). Observe that the order in which pictures will be dropped from a given set $pics_X(q)$ depends only on local information. This order can therefore be precomputed in advance. Since nodes in DTNs spend a lot of time between encounters when they apparently remain idle, there is enough time to compute the order in which pictures are to be dropped. The dropping policy pre-marks some content as deleted to create enough free buffer space to accept content from newly encountered nodes, and creates a single linked list in the order these pictures are to be removed. During an encounter, marked content is replaced if space is needed. After an encounter, the dropping policy pre-computes the order of deletion again. Locally generated content triggers periodic recomputation of the dropping order. The above algorithm implicitly assumes that the total amount of content delivered to a node during an encounter is not a significant fraction of its storage capacity. Hence, the odds that the newly received content is a better candidate for dropping than the pre-marked content is low. These odds are further reduced by the fact that the forwarding policy prioritizes content such that the most "useful" content is forwarded first. This mechanism is described next.

In case CAP needs to make dropping decisions on the fly upon a contact, an efficient implementation of computing dropping order can be made. Equation 3 suggests that pictures can be dropped at an ascending order of $\psi(x)/s(x)$, that is, the least diverse content first. When a particular picture, say y, is dropped, $\psi(x)$'s of remaining items are updated as follows: subtract $\|x-y\|^2$ from $\psi(x)$ (and normalize by the size of the collection; the same is added when a new picture is added to the set). Then, the remaining pictures are again sorted to find the candidate for the next drop. This can be implemented by a min heap that returns elements with minimum $\psi(x)/s(x)$.

C. A Prioritized Replication Policy

In DTNs, upon a contact, a node decides which messages it needs to replicate to the other peer, in some particular order. The term replication is slightly different than traditional forwarding. Here, the sending node *retains* the copy of the message that it transfers to the peer. This allows the same message to be transferred onto another node, opportunistically increasing the chance that at least one of these messages would be eventually delivered to the destination. CAP follows a simple replication policy "most diverse content first", that is, pictures that maximize diversity $\Psi(I)$ of the receiver's collection should be replicated first. In the following, we use the terms transfer, forward and replication interchangeably.

Hence, when node X meets Y, it needs to know exactly what pictures node Y currently holds for a given query so that X does not send similar content to Y. In other words, for each query, q, node X should forward to Y the pictures that maximize $\Psi(pics_Y(q))$ at Y. A naive approach could be that Y sends the feature vectors of all pictures in $pics_Y(q)$ to X for each query q. Hence, X could choose those pictures that, if forwarded, will maximize $\Psi(pics_Y(q))$. Intuitively, these pictures are the most distant in its vector space from Y's current picture set. Obviously, exchanging vectors for all stored pictures is costly, especially when the vector size is large and the number of pictures is many. This is also computationally expensive to compute distances from all pair of pictures. Instead, each node partitions its picture collection into clusters.

1) Clustering Pictures: Every node, X, clusters each of its picture collections, $pics_X(q)$ (one per outstanding query) and computes the centroid of each cluster, called a *pivot*. Hence, only pivots need to be exchanged. The above clustering is a function of only local information on the node, and hence can be done in free time in between encounters (i.e., before the node actually gets in contact with another). The clustering operation is done for each query in the query table. For a given number of pivots k, an optimal position for pivot points would be such that the sum of distances from non-pivot points to the nearest pivot is minimized over all other possible choices of pivot locations. This problem, known as k-mean clustering and reportedly NP-Hard, can be approximated by Lloyd's algorithm [4]. Lloyd algorithm starts with a random k pivots and then iteratively adjusts pivot locations based on assignment of pictures to the their respective nearest pivot. The clustering stops when the diameter of each cluster reaches within some threshold. The query table at each node stores the list of computed pivots, called $P(q)$, for each query q. These pivots are computed over all pictures, $pics_X(q)$, at node X.

2) Priority Order for Forwarding: Once pivots are computed, CAP's forward ordering of pictures matching query q is fairly simple. When two nodes meet, they exchange their pivot vectors for all queries. Let $P_X(q)$ and $P_Y(q)$ be the pivot vectors of nodes X and Y respectively, for query q. After these vectors are exchanged, node X computes the possible

diversity that a picture x (matching query q) can introduce to the existing picture set at Y. This is computed as the average square distance from the picture to the pivot vectors:

$$\psi(x, P_Y(q)) \quad = \quad \frac{1}{|P_Y(q)|} \sum_{y \in P_Y(q)} \|x - y\|^2 \qquad (4)$$

The picture that is farther away from the corresponding pivots of Y introduces greater diversity than other pictures with smaller distances. This is however obtained at the cost of transferring the picture itself to the peer node, which costs (in terms of energy or bandwidth occupancy) in proportion to the size of the picture (the length of the message in bytes). Therefore, whichever picture produces the largest gain per byte, $\psi(x, P_Y(q))/s(x)$, is given the highest priority. When the pivot set $P_Y(q)$ is well understood from the context, we drop $P_Y(q)$ from the diversity expression (Equation 4), to write $\psi(x)$.

Algorithm 1 replicate-messages(Contact $c : X \rightarrow Y$)

```
exchange query tables, i.e., name(q)'s
for each query q in query table do
    send P_X(q) to Y, receive P_Y(q) from Y
    populate pics_X(q)
    P(q) = P_Y(q)
    if P(q) = ∅ then
        P(p) = arg_{x,y} max ||x − y||, x, y ∈ pics_X(q)
    end if
end for
while connection c persists do
    pick the next query, q, in RR or WFQ manner
    set p = arg_{x∈pics_X(q)} max ψ(x)/s(x)
    send-picture(p, c)
    if picture p is transferred then
        P(q) = P(q) ∪ {p}
    end if
end while
```

3) A note on optimization: As an optimization, when computing pivots, we in fact compute them based on clustering the *union* of sets $pics_X(q) \cup pics_X^{past}(q)$, where the latter set denotes the pictures that match query q that have been previously forwarded by X to other nodes and erased. There is a good chance the query sink will receive these pictures. Hence, this retention of "previous memory" prevents forwarding semantically redundant objects to a destination if they arrive at the forwarder at different times. Otherwise, if the forwarder forgets such evicted content, it can end up subsequently forwarding similar content to the same destination, hence contributing to needless redundancy. To remember set $pics_X^{past}(q)$, When a message is deleted that has previously been forwarded to another node, CAP retains the feature vector of the deleted picture. These vectors can be eventually removed from $pics_X^{past}(q)$, although, in the current implementation, we do not expire such vectors. Feature vectors are considerably smaller in size than the original content. Therefore, they do not produce much storage overhead.

D. Additional mechanisms and Implementation issues

For more efficient replication, PhotoNet performs some additional work. While replicating onward, each picture contains a list of nodes that the picture has already passed through. In that case, the picture would not be replicated onto the same node again. Each node also maintains a list of picture IDs (not feature vectors) that have been delivered to the collection point for a given query. This list is exchanged when two nodes meet and is propagated into the network. When the list is updated upon a contact, the corresponding pictures are exempted from being replicated anymore since they are already delivered. They can still remain in the picture store, because some future query may look for them. If not, they can be deleted from the store.

CAP realizes its own replication and dropping policy as it maintains its content. It can be implemented in one of two possible ways. The first one is to augment the underlying routing protocol with a callback routine to be called when the router sends or drops messages, so it chooses the next message for transmission or dropping correctly. Another approach is to implement CAP as an application overlay. Thus, when nodes meet, they talk to their CAP-instances. This design requires the underlying routing layer to forward all received messages to the application layer, enabling CAP to implement its own routing and prioritization policies on top. We use the latter approach.

E. Handling Noise and Outliers

CAP prefers pictures to be scattered across the vector space allowing more dissimilar content to pass through. This is very much befitting the objective of maximizing event coverage. However, it makes CAP vulnerable to noise and outliers because noise and outliers may be different from usual pictures, or may be taken at locales from which no other pictures are reported. CAP identifies them as dissimilar objects and assigns higher priority to them in the transmission queue. This can be remedied penalizing transmission of messages from singleton clusters (i.e., clusters containing only one message).

IV. Evaluation

We evaluate PhotoNet in a post-disaster situation assessment scenario. We use ONE simulator [5] to emulate a post-disaster rescue operations based on a previously published post-disaster mobility (PDM) model [6]. Admittedly, reconstructing a realistic situation is hard in a simulator. Instead, we try to capture key aspects and elements of the scenario and compare all competing approaches on the same grounds. We compare our proposed picture delivery service with two other DTN protocols, namely Prophet and SprayAndWait (henceforth we refer to as Spray), and observe how our scheme improves performance in terms of coverage-related performance metrics.

DTN routing protocols are of two main types: flooding-based and quota-based. Prophet is a flooding-based scheme that computes path metric in terms of probability of delivery by using histories of encounters in a mobile DTN. It directs message propagation toward the direction where probability of delivery only increases. Spray is a quota-based protocol that limits the number of replica of a given message. Every message starts with a replica count header, which is halved at every replication upon a contact until the count becomes

1, when the message can only be delivered directly to the destination without being replicated anymore. Both Prophet and Spray use drop-tail policy of dropping messages (drops the earliest message first). We chose these two protocols, one from each type, to compare with PhotoNet.

As an instance of a functional prototype, we also implemented the service on a small mobile testbed with a few handheld Android phones. Since large scale communication as envisioned by the application scenario can hardly be emulated in such a small testbed, we better do communication experiments on simulation and show results of node level experiences from testbed. We also set a few parameters of the simulation from testbed measurements (Section IV-C). We leave a real large scale deployment of PhotoNet to future work.

We split our evaluation section into three parts. First, we detail our simulation environment, then present performance results and comparison, and finally describe our experiences of building the service on a mobile platform.

A. Simulation Environment

There are two key elements of the simulation: i) simulating the mobility of agents, ii) simulating the generation of events.

1) Mobility Model: PDM models movement of various agents, mainly humans and vehicles, in a post-disaster recovery situation. PDM is implemented on top of ONE's Map-Based Mobility Model that uses map data of roads and streets. It first places a few neighborhoods scattered on the map and then puts houses and survivors in those neighborhoods. It then randomly places a specified number of relief camps, police stations, command stations and other entities on the map (provided in a configuration file). After that, it deploys moving agents of four major types: center to center (recurrent back-and-forth motion of supplies between centers), rescue workers and volunteers (localized random motion within a certain neighborhood), cyclic patrols (recurrently patrolled paths through multiple neighborhoods), and emergency responses (dispatch of vehicles from a center to a random destination and back). All static and mobile agents are equipped with wireless routers capable of running DTN protocols. Figure 2 depicts a small city map (that comes with ONE) to describe an urban disaster area.

The underlying mobility model forms some kind of an aggregation tree for collecting pictures. Rescue workers and volunteers, instrumented with cameras, move in neighborhood areas and shoot pictures, and occasionally report to some neighboring relief camps. Supply vehicles and police patrols visit neighborhoods. There are a few vehicles that visit the main command station and the relief camps at each neighborhood. The later moving nodes work as "data mules" that collect pictures from neighborhoods and deliver to the command station. In our deployment, we have 100 houses and 100 rescue workers in 5 neighborhoods and 5 relief centers with 5–10 supply vehicles. We have a few, say 2–5, data mules to connect neighborhoods with the command station. These mules constitute communication bottlenecks in picture collection.

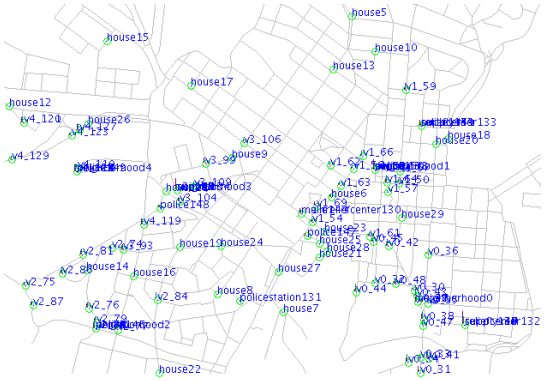


Fig. 2. The city map used by the simulator

2) Generation of Events and Pictures: PDM simulates movements of agents, but not events. We extend PDM to incorporate event generation and event reporting activities. To model events, we randomly choose a few locations on the map as event locations. These points are preferably at neighborhood locations where occurring of events is likely and where agents frequently visit. We then associate each location with a certain set of events that "occurred" at that location. Events, in this case, correspond to instances of damage, collapse, fires, or other hazards.

Each event is supposed to have a distinct appearance. Since we do not have real events happening in the simulation, we pre-take a set of real pictures of a few distinct landmarks and objects around our campus, and map each landmark to a certain event. We take several pictures for each landmark from different angles and zoom levels to mimic the reality that several users take the same picture differently. Figure 3 shows five sample pictures of event "fallen rocks". We then assign specific sets of pictures of landmarks to locations. We argue that there could be some events that are seen by many observers (popular events), whereas others may be less popular. To emulate this effect, we use Zipf distribution to determine the number of different images assigned to certain events for each neighborhood. We assign a popularity index (1 means highly popular) to each event and generate $\lceil \frac{n_{max}}{i^2} \rceil$ number of pictures for an event with an index i. We used $n_{max} = 50$.

When an agent happens to pass or stop by a certain event location, it randomly chooses one or more pictures from the pre-arranged set, as if it just "took" pictures of this event and a message is created in the network. Once taken, the picture is deleted from the set so that no other agents report exactly the same picture. Once stored, a certain preprocessing time needs to elapse (compressing, reducing the dimension and extracting features from the picture; Section IV-C presents the timing results) before the picture is eligible to be forwarded onto others upon contacts.

We use color histogram and KL-divergence distances for image-features. Figure 4 shows the cumulative distributions of KL-distances among pictures used in the simulation for a total of 32, 64 and 128 color bins. We see that dissimilar pictures are further away from similar pictures in histogram space, which enables CAP to cluster them separately, if image-features are

Fig. 3. Five similar pictures of an event "fallen rocks"

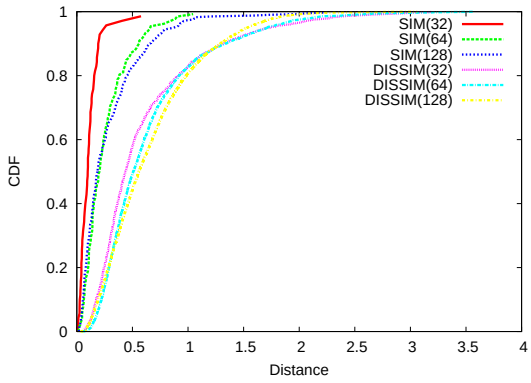


Fig. 4. CDF of KL-distances among similar and dissimilar pictures

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Storage capacity	1–10MB	Picture size	100KB
Picture proc time	4s	Conn estab. time	2s
Trans radius	20m	Trans rate	250 KB/sec
Spray quota	50	Prophet const	(0.75, 80) [7]

used. We used 32 color bins, which makes meta-data overhead per picture quite small ($8 + 8 + 32 \times 4 = 144$ bytes).

B. Performance Evaluation

Next, we evaluate PhotoNet with two DTN routing protocols, Prophet and Spray. Table I shows default parameters used in the simulation. We are interested in results in an operating condition when resources are so limited that usual network performance is very poor. For example, in almost all experiments, if not otherwise shown explicitly, picture delivery ratio, the fraction of total pictures delivered over total generated, is very low, around 10%-20%. PhotoNet intends to serve as many diverse pictures as possible out of these very limited delivery. We focus on evaluating the performance of the default query (collecting all images from all sources).

We compare performance of PhotoNet to other protocols under resource constraints. To see that the underlying DTNs are constrained, consider a network of 5000 nodes (e.g., relief workers and volunteers in a disaster recovery scenario), generating 100KB pictures from head-mounted cameras at the rate of approximately 50 pictures per hour. If a data mule eventually delivers these pictures to the destination every four hours, the mule will need to have a storage capacity sufficient for $4 \times 50 \times 5000 = 10^6$ pictures, or about 100GB. This amount of storage is challenging, considering that the mule may be just a mobile handheld. To keep simulation time low, we do not generate thousands of pictures. Instead, we reduce the number of pictures as well as the assumed device storage capacity proportionally. To this end, we generate nearly 25 pictures per hour for a network of 100 nodes, reducing the above storage requirements to 10MB (to hold 100 pictures in lieu of 10^6). On the same ground, in order to scale communication capacity, we set smaller radio transmission range (20 m) as well as low link transmission rate (sometimes 50KB/sec).

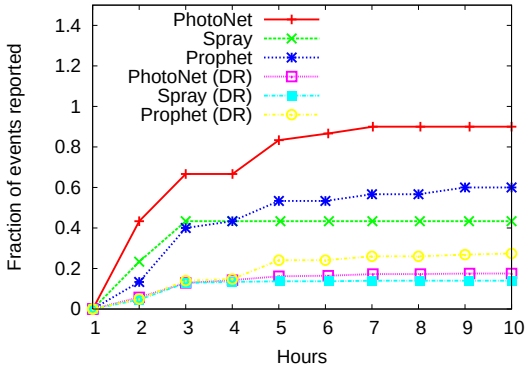
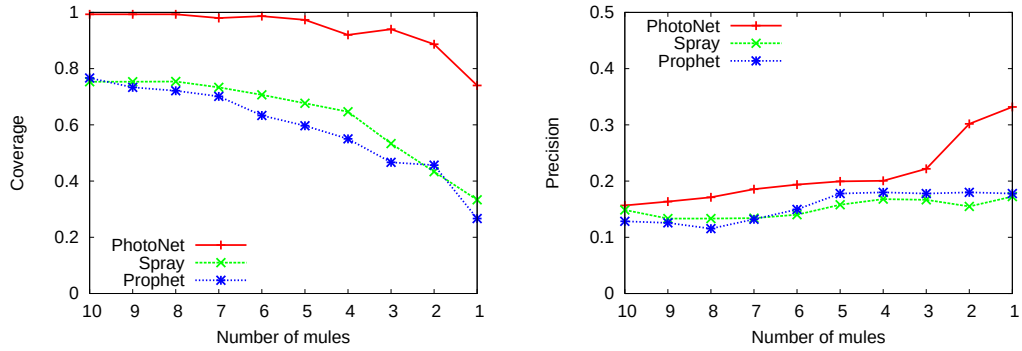


Fig. 5. Hourly event collection for PhotoNet, Prophet and Spray

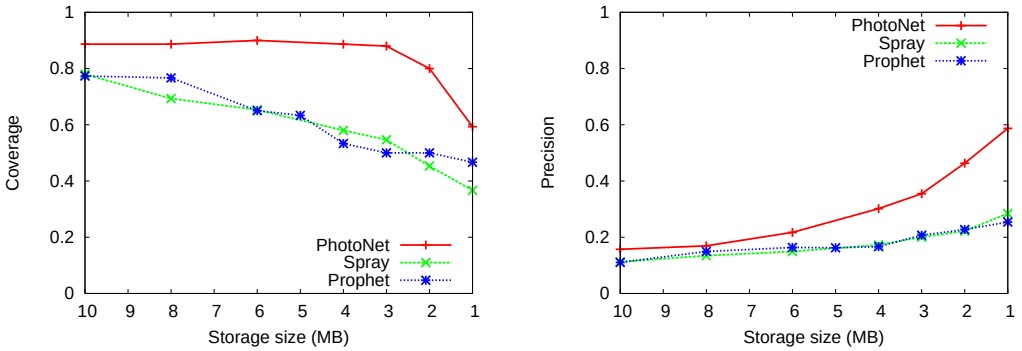
PhotoNet intends to collect as many different events (i.e., pictures) as possible. We define two performance metrics, *event coverage* and *precision*. Event coverage computes the fraction of total *distinct* events that have been successfully reported at the sink to the total number of distinct events generated. We say that an event is *reported* if at least one picture pertaining to that event has been delivered to the sink. Precision measures what fraction of delivered pictures were unique, that is, the first picture that contributed to reporting an event. PhotoNet aims at achieving higher event coverage as well as higher precision. Note that higher precision means lower overhead.

Figure 5 shows the fraction of total events reported to those generated as a function of time for PhotoNet as well as for Prophet and Spray within some defined *deadline*, reportedly 10 hours. For this experiment, we generate all pictures at sometime around 1 hour, and see what fractions are delivered by the deadline. We observe that the PhotoNet reports more events (30–40% more) than regular Prophet and Spray in a any given time. We also plot delivery ratio, the fraction of total pictures delivered to total generated, in every hour. It is observed that the delivery ratio of PhotoNet is slightly low (compared to Prophet), still its event coverage is higher than others. It is also to note that although delivery ratio is within 20%, PhotoNet delivers nearly 80% of total generated events. This is due to the prioritization scheme applied by PhotoNet.

The number of data mules connecting neighborhoods with the command station affects the connectivity of the deployed area, hence the picture collection efficiency. Figure 6(a, b)



(a) Coverage (Mules) (b) Precision (Mules)



(c) Coverage (Buffer) (d) Precision (Buffer)

Fig. 6. Event coverage and precision at (a, b) varying number of mules, (c, d) varying storage capacity

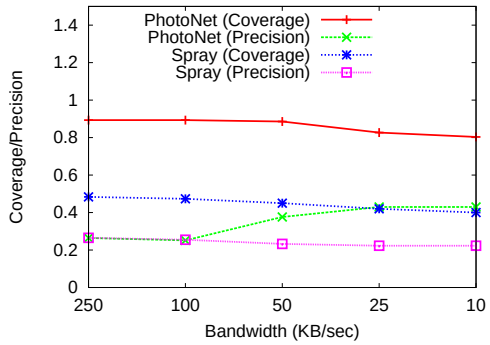
show event coverage and precision at varying number of data mules at storage capacity 5MB. We see that when the number of data mules is large, event coverage is moderately high for all protocols and it then declines for other protocols except for PhotoNet when fewer mules are deployed. Fewer number of mules causes less carrying capacity, thus more events failed to be reported to the sink. Since PhotoNet uses CAP (content-aware prioritization), it exploits connection resources and delivers diverse content first. Therefore, its event coverage does not decrease much with decreasing number of mules. But at certain point (say at 1 mule), poor connectivity dominates other constraints and PhotoNet's event coverage also declines. As relay capacity becomes weak, PhotoNet's precision rises whereas others remain quite the same. This is because CAP chooses diverse pictures to go first and consequently raises the ratio of the number of distinct pictures delivered to total delivered.

Figure 6(c, d) show the same set of results when storage capacity is varied. In case, storage becomes a bottleneck, more and more pictures would be dropped from nodes. Dropping pictures in some discriminate fashion would simply drop different events altogether possibly serving only most replicated popular ones. In contrast, PhotoNet gives priority to diverse content among stored pictures and drop most redundant (less diverse) content first. It thus holds as many different events as possible. Figure 6(c) depicts that at a high storage all protocols start at a good event coverage, but for others event coverage eventually declines as storage becomes scare, but PhotoNet still offers higher event coverage as well as higher precision. At some extreme poor state though (at 1MB storage that can hold only 10 pictures), it also suffers.

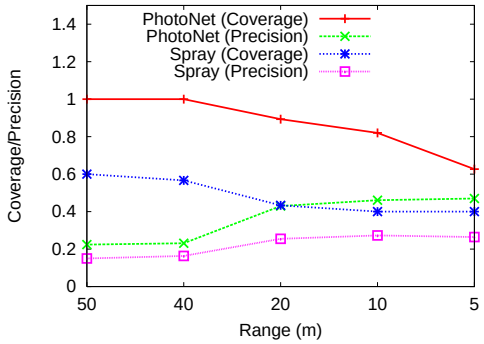
Figure 7(a) shows event coverage and precision at varying link bandwidth and transmission range. In these cases too, PhotoNet outperforms others both in coverage and in precision. We observe that despite bandwidth gets low, event coverage is not affected that much. This is because, in our mobility model, there are a few static points where nodes stay for a while and those contacts are relatively large. If not otherwise constrained by the storage capacity, this allows nodes to exchange their pictures where only prioritization does not help much.

C. A Testbed Implementation

We implemented PhotoNet as a small but functional prototype on a mobile testbed with a few Android phones. In our testbed implementation, phones are allowed to take pictures and communicate with other devices for exchanging pictures. To test the service, we visited various places and shot pictures using different phones. All pictures were tagged with GPS coordinates of places they were taken. We then set these phones to exchange pictures in an emulated DTN environment. This was done manually by repeatedly connecting and disconnecting pairs of devices via a special-purpose application GUI. We were interested in seeing that after a several rounds of exchange, the devices end up having pictures that are considerably diverse, when each device is allowed to store only a limited number of pictures.



(a) Bandwidth



(b) Transmission range

Fig. 7. Coverage/precision at varying link bandwidth and trans range

TABLE II
VARIOUS TIMING VALUES ON ANDROID PHONES

Component	Average time
Taking picture, compress and store	2264.55 ms
Computing color histograms	1981.0 ms
Discovery and connection establishment	2656.76 ms

Other than serving as a functional prototype of the PhotoNet service, the phone based implementation gives us node-level measurements (for example, time to extract image features). The implementation also helped determine some of the simulation parameters used in the earlier section (e.g., connection setting time). In this section, we report on a few timing characteristics that were not captured by the network simulator presented in earlier. We present all computed timing values in Table II. These activities are mainly offline operations that occur when nodes do not communicate with others.

Once a picture is captured in our implementation, it is compressed into JPEG form, tagged with GPS location and then stored in the local storage. This compression is required because raw bitmap data is usually very large (average 1.21MB). Compressed images average 135KB, nearly 11% of the raw data. Once compressed, image-features (i.e., color histograms) are extracted from the captured pictures. This computation occurs only once. The resulting vector is then added to the application-level message header as meta information. We implemented feature extraction on an Android phone and computed the average time required to populate color histogram features. We show the average histogram computation times for pictures with different sizes in Table III. Finally, the user is prompted to name the picture in the hierarchical content tree structure. The picture is now ready to be served against a query.

TABLE III
AVERAGE FEATURE EXTRACTION TIME FROM PICTURES

Picture size	Avg. time (ms)
360x480	719.7
600x800	1981.0
1200x1000	4898.2

In DTNs, nodes need to discover other nodes to do opportunistic communication. Once discovered, the associated devices establish connection between them. This discovery and connection establishment take some time, based on distance between devices, surrounding environmental conditions and other factors. We measured the average time elapsed in discovery, pairing, and connection establishment time over bluetooth (the communication medium used in the current implementation) at each contact between a pair of Android phones. We manually pair each pair of phones beforehand and keep them listening on a connection socket. Once a pair of devices establishes a connection between them, the timing is recorded. We used this time as a connection establishment overhead in each contact in the simulation.

There is another important offline computation overhead in PhotoNet, which is to compute pivots. Pivots (per query) produce a summary of pictures on a node and are computed on all stored or ever replicated pictures (for a given query). We show the average clustering time for a set of pictures in Figure 8. It is shown that as the number of pictures increases, clustering time increases, somewhat non-linearly: more time for larger image set. This computation although costly happens only when nodes are otherwise idle.

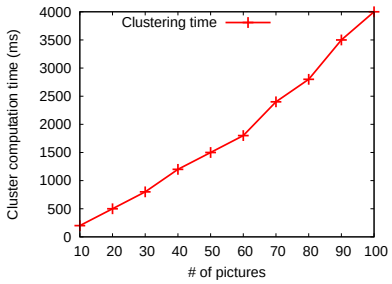


Fig. 8. Avg clustering time at varying # of pictures on an Android phone

V. RELATED WORK

PhotoNet runs on DTNs. Due to intermittent connectivity in DTNs, messages are usually replicated to ensure better delivery. Based on the level of replication, DTN routing protocols are of two types: flooding- and quota-based. Prophet [7], MaxProp [8], Delegation routing [9], RAPID [10] are a few flooding based protocols to name, whereas Spray [11], EBR [12] and IC-Routing [13] are quota-based. The above previous protocols are all content-agnostic.

PhotoNet belongs to content-aware networking services. Content-based data forwarding has been explored in previous literature. In [14], a content service model was proposed with

content brokers, so that content can be stored and disseminated efficiently based on users' requests. CNF (cache-and-forward) [15], proposed a similar idea where a few gateway nodes at the edge of the network store data items so that mobile agents can obtain their data once they are online. Other work addressed content caching [16] and content aware routing [17], [18]. CCN [3] introduced networking of named content, in place of IP networking by hosts. The above schemes, despite their efficient retrieval and dissemination of content, are not content-aware in that they do not explore *semantics* of content.

Data fusion literature described semantic-aware content fusion methods. Semantic fusion has usually two phases: knowledge base construction and pattern matching [19]. At first phase, a suitable abstraction for representing semantic information is chosen, which is then used in second phase for matching and fusing relevant attributes. This fusion runs in-network inference processes so that nodes only exchange semantic interpretations. In [19], authors applied semantic fusion for target classification. Another work [20] integrates sensor data into formal languages, and then matches data with some stored knowledge base based on the hypothesis that data represented by similar languages are semantically similar. Semantic streaming [21] allows users to formulate queries over semantic values without specifying data or operations. SONGS architecture, proposed in [22], uses declarative queries and converts queries into service composition graph. Similar to semantic fusion, our proposed technique, CAP, also has two phases: vector representation of content for semantic abstraction and content matching by computing distances in vector space. Unlike other fusion methods, CAP does not perform vertical integration by fusing items, instead it does vertical comparison. CAP compares data content in storage for semantic similarity and prioritizes content for transmission and dropping so that a certain quality objective is met.

VI. Conclusion and Future Work

In this paper, we propose and build a picture delivery service, PhotoNet, for resource-constrained DTNs. PhotoNet addresses the challenge that when pictures are generated and serviced by a mission-driven network, there could be significant overlap between their content. To improve resource efficiency PhotoNet computes semantic similarity and prioritizes content accordingly so that delivered messages maximize event coverage.

The paper is a first step that presents a proof-of-concept investigation towards a more comprehensive study of content-aware protocols for different resource-constrained environments, applications, and content types. The investigation suggests that the approach could have great impact on improving resource efficiency in appropriate scenarios. Future work of the authors will focus on exploring the limits of usability of the approach, as well as on generalizing it to heterogeneous content, more complex application goals, and multiple concurrent applications. Ease of network customization to application goals will also be investigated.

Acknowledgements

Research reported in this paper was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*. Elsevier, 2007.

[2] J. N. de Souza and R. B. (Eds.), *Managing QoS in Multimedia Networks and Services*, ser. IFIP Advances in Information and Communication Technology. Springer, 2000.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of CoNEXT*, New York, NY, 2009.

[4] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, pp. 637–676, 1999.

[5] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for dtn protocol evaluation," in *Proc. of the 2nd International Conference on Simulation Tools and Techniques*, Brussels, Belgium, 2009.

[6] M. Y. S. Uddin, D. Nicol, T. Abdelzaher, and R. Kravets, "A post-disaster mobility model for delay-tolerant networking," in *Proc.of Winter Simulation Conference*, Austin, TX, December 2009.

[7] O. S. Anders Lindgren, Avri Doria, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2003.

[8] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. of INFOCOM)*, April 2006.

[9] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proc. of ACM MobiHoc*, 2007.

[10] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. of ACM SIGCOMM*, 2007, pp. 373–384.

[11] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05)*, Philadelphia, PA, 2005, pp. 252–259.

[12] S. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in DTNs," in *Proc. of IEEE INFOCOM*, 2009.

[13] M. S. Uddin, H. Ahmadi, T. Abdelzaher, and R. Kravets, "A low-energy multicopy inter-contact routing protocol for disaster response networks," in *Proc. of IEEE SECON*, 2009.

[14] B. Subbiah and Z. A. Uzmi, "Content aware networking in the internet: issues and challenges," in *Proc. of ICC*, Helsinki , Finland, 2001.

[15] 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," in *Proc. of Innovations in NGN: Future Network and Services*, 2008.

[16] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area Web cache sharing protocol," *IEEE/ACM Trans on Networking*, 2000.

[17] E. Kang, S. Lee, and M.-S. Park, "Efficient prioritized service recovery using content-aware routing mechanism in web server cluster," in *Networking - ICN*. Springer Berlin / Heidelberg, 2005, pp. 297–306.

[18] S. Eichler, "MDRP: A content-aware data exchange protocol for mobile ad hoc networks," in *Proc. of International Symposium on Wireless Communication Systems*, 2007.

[19] D. S. Friedlander and S. Phoha, "Semantic information fusion for coordinated signal processing in mobile sensor networks," *Int. Journal of High Perf. Comput. Appl.*, vol. 16, pp. 235–241, 2002.

[20] D. Friedlander, *Semantic information extraction*, S. S. Iyengar and E. R. R. Brooks, Eds. CRC Press, 2005.

[21] K. Whitehouse, J. Liu, and F. Zhao, "Semantic streams: A framework for composable inference over sensor data," in *Proc. of EWSN*, Zurich, Switzerland, 2006.

[22] J. Liu, E. Cheong, and F. Zhao, "Semantics-based optimization across uncoordinated tasks in networked embedded systems," in *Proc. of ACM international conference on Embedded software*, Jersey City, NJ, September 2005.