InfoMax: An Information Maximizing Transport Layer Protocol for Named Data Networks

Jongdeog Lee, Akash Kapoor, Md Tanvir Al Amin,
Zeyuan Zhang, Radhika Goyal, Tarek Abdelzaher
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
{jlee700, akapoor5, maamin2, zzhan116, rgoyal3, zaher}@illinois.edu

Zhehao Wang
REMAP
University of California at Los Angeles
Los Angeles, California 90095
zwang@tft.ucla.edu

***Abstract*—The advent of social networks, mobile sensing, and the Internet of Things herald an age of data overload, where the amount of data generated and stored by various data services exceeds application consumption needs. In such an age, an increasingly important need of data clients will be one of data sub-sampling. This need calls for novel data dissemination protocols that allow clients to request from the network a representative sampling of data that matches a query. In this paper, we present the design of a new transport-layer dissemination protocol, called InfoMax, that allows applications to request such a data sampling. InfoMax exploits the recently proposed named-data-networking (NDN) stack that makes networks aware of hierarchical data names, as opposed to IP addresses. Assuming that named objects with longer prefixes are semantically more similar, InfoMax has the property of minimizing semantic redundancy among delivered data items, hence offering the best coverage of the requested topic with the fewest bytes. The paper discusses the design of InfoMax, its experimental evaluation, and example applications.**

I. Introduction

This paper[1] develops a new transport-layer communication abstraction and underlying data dissemination protocol that allow a consumer to request a *sampling* of a producer's data at a *configurable granularity*. The protocol is implemented on top of a named-data-networking (NDN) stack. It represents an example of a new transport paradigm centered around lossy information transfer that aims to substantially reduce transmitted data volume, while meeting application summarization needs.

The work is motivated by the proliferation of sensors that are connected to the Internet, and the advent of social networks that democratize information broadcast, propelling us into a world of data overload at an accelerating rate. Indeed, over 90% of the world data was produced within the last decade [1]. As the amount of digital data grows exponentially, a gap arises between the over-abundant supply and actual application demand. The growing disparity between the amounts of data generated and what suffices to meet application needs suggests that an increasingly important function in the future will be one of data sub-sampling (as a means of summarizing large data).[2] Different data uses might have different sub-sampling needs for the same data set. The current Internet architecture places the sub-sampling burden on the information client. The paper shows that *information centric networks* significantly change the equation by facilitating the development of generic lossy transport, that has the effect of sub-sampling data in a manner that achieves minimum information loss, while reducing information volume in a client-controlled fashion. We expect that this protocol will be of increasing importance to a growing range of data-centric applications.

Our protocol is implemented on top of the *named data networking* (NDN) protocol stack [2]. NDN, an instance of information-centric networking, is a network layer that assigns hierarchical names to data objects instead of hosts. This architectural decision has two important implications from the perspective of InfoMax design. First, the network and transport layers are aware of application-level object boundaries. This is because objects have explicit names that the network is aware of. Second, since a hierarchical naming scheme is used, inferences can be made about semantic overlap between named objects based on the distance between the corresponding names in the name tree. These two insights allow us to develop an information-loss-minimizing protocol for data sub-sampling.

Unlike the IP paradigm, which is *push-based* (where a sender decides what a receiver gets), InfoMax lies atop of NDN, which is *pull-based*. The receiver pulls data from the network by name. (Note that, in NDN, since data are named, it does not matter who the sender is.) Hence, InfoMax offers a primitive for requesting a data set specified by a name prefix. The prefix names the root of a content tree with the understanding that a sampling of the content tree is to be transmitted to the requesting node. The requesting node does not need to know what content is stored in the tree and does not know the tree name space. It just wants the data under the tree

[1]Work on this paper was supported in part by NSF grant CNS 13-45266 and CNS 10-40380.

[2]In this paper, we use the word "sampling" or "sub-sampling" in the conventional general sense of selecting representative examples of objects from a larger set of similar objects. For instance, in a tourism application one might want to select examples of good and bad reviews of some hotel. In a building management application, one might want to select examples of afternoon temperature readings in a building.

at a configurable level of summarization (i.e., sub-sampling). Since it is the receiver of the data that controls the amount of information transfer, the receiver can pull less content or more content, until the receiver's content needs are satisfied. Hence, besides specifying the name prefix that decides the data set, the receiver controls the degree of summarization by specifying how much information is actually pulled from the named tree.

The novelty of InfoMax lies in the order that objects belonging to this tree are transferred. InfoMax assumes that objects that share a longer prefix within the tree have more semantic overlap. Hence, the marginal utility of sending an object that shares a longer prefix with a previously transmitted one is *less* than the marginal utility of sending an object that shares a shorter prefix. This leads to a *shortest shared prefix first* transmission order for objects in the requested tree. This order is shown to maximize the marginal utility of transmitted objects and hence the total accumulated utility for a given transmitted data volume. Said differently, a transmission can be stopped at any time, short of transmitting the whole tree, while ensuring the property that the information loss resulting from early termination is minimized over all possible object transmission orders. Since, in a pull-based architecture, it is the receiver that controls when transmission stops, the receiver gets to decide when they've had data at a fine-enough granularity. The idea is reminiscent of receiver-driven multicast [3], implemented in an application-independent fashion.

InfoMax extends a previous transport layer abstraction, called the *Information Funnel* [4], proposed earlier by the authors. While the latter was geared for data *collection* by a single consumer from multiple producers, InfoMax is geared for the complementary case of data *dissemination*, where multiple consumers pull data from the same producer, possibly at different levels of summarization. Together, the Information Funnel and the InfoMax dissemination protocol may constitute the two building blocks of many data-intensive services. The funnel will collect data into repositories from multiple sources, whereas InfoMax will serve it to multiple eventual consumers at different degrees of detail. We also significantly simplify the data ordering algorithm, compared to the algorithm used by the Information Funnel, while ensuring that it maximally benefits from network caching within the NDN Layer. The protocol was implemented and tested on the NDN testbed. At the time of writing, the testbed includes more than 15 institutions, spanning three continents (although our tests were performed in the continental US). Two applications are illustrated; one from the domain of tourism and the other from the domain of social network search. Results demonstrate the advantages of using InfoMax in terms of achieving a configurable receiver-decided trade off between data volume received and degree of detail. We also show that the scheme offers a relatively low overhead and that it leverages native NDN in-network caching.

The rest of the paper is organized as follows. The design and implementation of the new protocol are described in Section II. Section III describes the experimental setup and explains the evaluation results on a nation-wide testbed. Section IV describes two prospective application examples. We discuss related work in Section V and conclude this paper with a summary and discussion of future work in Section VI.

II. Design and Implementation

In this section, we discuss the InfoMax abstraction, design decisions, and implementation on top of the named-data-networking stack.

A. The InfoMax Information Summarization Abstraction

The InfoMax protocol is designed to sample data of applications that store and serve significant amounts of largely homogeneous, and partially redundant data. An example application class that fits this category is social networks. For instance, Twitter serves data (collected from a large population) as JSON objects; each object is a tweet together with its metadata. Similarly, Instagram and Flickr serve images; each image includes a picture file and EXIF metadata. These data sets are large and syntactically homogeneous, but amenable to hierarchical clustering based on semantic content. For example, recent work suggested a hierarchical clustering framework for tweets [5], that significantly speeds up indexing to retrieve tweets that match arbitrary keywords. The hierarchy arises because some tweets are more similar than others, as can be measured by various text similarity metrics. At the top of the hierarchy are tweet clusters that have no text overlap. Subsequent levels introduce tweets with progressively more overlap with their siblings. If a name space echoed the above hierarchy, tweets whose names have a longer shared prefix would be more similar.

Cyber-physical systems offer another application class that gives rise to syntactically homogeneous data objects that semantically fall into hierarchical structures. For example, a campus building management system might collect data on temperature in different campus spaces. Such data may be organized hierarchically by building, floor, room, and temperature sensor [6]. Clearly, data from sensors in the same room would be more similar than data from different floors or buildings. If the data were named hierarchically as described above, items that share a longer name prefix would generally be more correlated (being from the same or close physical spaces). With the rise of projects such as the Internet of Things, and the increased ubiquity of sensory instrumentation, the amount of generated sensor data is going to increase making cyber-physical data sets more widely available on networks.

The InfoMax protocol is designed as a general protocol for requesting a sampling of such data sets that summarizes them at a receiver-driven degree of granularity. Name space design is clearly a key consideration when using InfoMax, in that it determines the output of summarization. The examples above illustrate that the name space in important categories applications lends itself naturally to a hierarchical organization that is suitable for InfoMax. The main API calls provided by the InfoMax producer and consumer, along with their uses, are summarized in Table I and Table II, respectively.

TABLE I: InfoMax Producer APIs

`InfoMaxProducer (Name prefix)`: Constructor for the InfoMax producer object, `producer`. It is called by an application to initialize the producer and specify a name prefix associated with this producer. The prefix specifies a content name space that the producer should make available on the network.
`add (Name postfix, Data data)`: A method of the producer, `producer->add` to add data objects to its tree. The call specifies the data object that needs to be added along with its position in the tree.
`delete (Name postfix)`: A method of the producer, `producer->delete` to delete the specified tree branch.

TABLE II: InfoMax Consumer APIs

`InfoMaxConsumer (Name prefix)`: Similar to the InfoMaxProducer, this is a constructor for the InfoMax consumer object, `consumer`. A prefix is specified that the consumer is interested in. It creates the receiving end for all content that matches this prefix.
`get ()`: This method, `consumer->get`, waits to receive the object from the network that match the prefix requested by the consumer (or timeout). The method may be called multiple times to receive the set of matching objects. The InfoMax protocol continues to retrieve objects until the application stops calling this method. The application at the receiver can continue to retrieve objects until the desired level of detail is met.

B. A Note on Optimality

The InfoMax protocol was designed to (i) minimize overhead, (ii) make the best use of the underlying NDN stack, and (iii) offer a general sub-sampling function that is application-independent. By sub-sampling, we do not mean returning an arbitrary subset of requested objects. What is the interesting about InfoMax is that it retrieves the subset that *minimizes* information loss (compared to receiving the entire set), given the fraction of retrieved objects.

Definition 1. The name space assumption: The fundamental assumption that InfoMax makes regarding an application's name space, is that data objects have hierarchical names that respect the following property: The longer the shared name prefix between two objects, measured in the number of shared name segments (i.e., tree levels), the lower the marginal utility of receiving the second object.

In accordance with the above assumption, objects are retrieved in a *shortest-shared-prefix-first* manner. Below, we offer a proof sketch of why this retrieval policy minimizes information (utility) loss.

Theorem: *Shortest-shared-prefix-first retrieval order minimizes receiver's information utility loss.*

Proof Sketch: Let us define *information utility* of an object at a receiver as a measure of value that the receiver gets from receiving the object. Consider a name prefix, advertised by a producer, that names a tree, T, containing a total of N data objects. Each object, $D_i \in T$ has an information utility, U_i, when received. Given the *name space assumption* mentioned above, U_i is a function of the length of the shared prefix between D_i and previously received objects. Let us denote the length of the largest such shared prefix by S_i. Hence, $U_i = f(S_i)$, where f is a non-increasing function. Shortest-shared-prefix-first implies that the sent object will minimize S_i over the remaining (unsent) object set, $R \subset T$. Hence, it will maximize $f(S_i)$. Since $f(S_i)$ is maximized for each sent object, it is possible to prove by induction over the number of received objects, K, that the sum $f(S_i)$ over the K objects received in InfoMax order is maximum among all possible sums of K objects. Hence, the utility sum over the remaining (unsent) object set, R, is minimum, which is defined as the *information utility loss* should transfer stop after K objects. Hence, InfoMax minimizes information loss for any K. □

Two observations should be made here. First, the above proof hinges on the assumptions that (i) U_i is *only* a function of S_i, denoted $U_i = f(S_i)$ in the proof, and (ii) the function f is non-increasing. The optimality result does not hold for more general utility functions. For example, it does not hold when marginal utilities are a function of the *identities* of received objects. Second, it is notoriously hard to figure out utility values in practice. Hence, it may be counter-productive to consider more nuanced utility models. Instead, we follow the Occam's Razor principle. Accordingly, the above proof sketch develops intuitions regarding optimality in the *simplest case* that is consistent with the name space assumption (labeled **Definition 1**, above).

C. The InfoMax Protocol

The InfoMax protocol [7], [8] is implemented on top of NDN. NDN offers an API to request single data objects by name. This API is called an *NDN interest*. A consumer can send an NDN interest to retrieve a named object from the network. NDN will route that interest to the nearest node (router or host) that has a matching object. The object is then returned following the reverse route. Routers on the reverse path cache objects that they see. InfoMax uses this mechanism to retrieve a content tree by repeatedly using NDN interests to retrieve the individual objects. The key contribution of InfoMax is its implementation of the shortest-shared-prefix-first retrieval order for objects in the requested tree. In this paper, we assume that content has a *single producer*. However, this content can be cached at multiple places in the network.

1) Enforcing the InfoMax Order: The first design question is: who should maintain the InfoMax shortest-shared-prefix-first order? When requesting a content tree that matches a prefix, the InfoMax consumer does not know what content exists in the tree. Hence, the consumer cannot request objects in the tree in the correct order. It is therefore the producer's responsibility to compute a retrieval order for the tree.

Native NDN allows consumers to send interests in partially specified name prefixes (e.g., the root of the desired content tree) with an exclusion filter that specifies objects (matching the prefix) that the consumer already received. Unfortunately, this approach does not specify which object to retrieve next from the tree.

A trivial design would have the producer keep track of how many objects were delivered so far to each consumer. When the next request arrives from a consumer, the next object matching that consumer's interest would be delivered. The design would require producers to keep per-client state, which undesirable. Furthermore, any design that requires recipients of consumers' NDN interests to know the *next* object to send in InfoMax order (e.g., based on previously received objects) would not work in the presence of network caching. Caches (unless they run InfoMax code as well) will not know the correct InfoMax order and hence may not serve the correct next object. To keep caches from having to implement InfoMax, consumers should always ask for the *full* object name, as opposed to a partially specified name prefix.

The requirement that producers maintain the shortest-shared-prefix-first list for their content (since they know their content best), together with the requirement that consumers ask for objects by their exact full name lead to a simple design where:

- *At the producer:* The transport entity sorts objects in the tree (under the exported name prefix) in a shortest-shared-prefix-first delivery order. This sorted order is recorded, broken into smaller objects, called *lists*. These lists are numbered and made available by the transport entity to be retrieved sequentially by consumers.
- *At the consumer:* Given a tree prefix, say $root$, to retrieve objects from, the transport entity requests the aforementioned numbered lists using default names, such as $root/list/1$, $root/list/2$, etc. (In our implementation, the default list names contain special characters not used in naming data objects, in order not to clash with data object names.) Once a list is retrieved, the transport entity at the consumer parses it for data object names, and requests the listed objects by exact name in the order they appear on the list.

The approach allows for both list objects and data objects to be cached. The cache need not run InfoMax code. It simply responds to interests in named list and data objects, either serving them or forwarding the interest further towards a producer.

The size of the lists turns out to be an important design parameter that can significantly impact communication performance. A small list size could lead to frequent exchange of lists in order to determine which data objects to retrieve next, thus leading to more network traffic. A large list size can increase overhead if the consumer does not need that many data objects in their tree summary. Since the network-layer packet in NDN is 8KB long, we fixed our list size to 100 object names, which is roughly 8KB in size. Hence, list objects take one full NDN packet. This is suitable for transfer of large numbers of data objects, which is the common case in the data intensive applications that motivate InfoMax. Whenever the consumer needs more data objects, it can ask for the next list and will get names of the next 100 objects.

Another important parameter to consider is the interest pipelining window size. That is to say, how many objects from a list should the consumer's transport entity request in parallel? This parameter is exported by the NDN layer. Its effect is empirically determined in the evaluation section.

2) Handling Dynamic Updates: The approach mentioned above is suitable for retrieving primarily static content. If the tree at the producer is updated often, two problems occur. First, extra overhead is paid in recomputing the lists that maintain the sorted order of objects in the tree. Second, if the lists are updated while some consumers are retrieving content, retrieval of inconsistent list versions may result in an incorrect data object retrieval order.

To reduce the overhead of recomputing the lists (as well as the number of list versions generated and hence potential cache pollution with these versions), the protocol controls the frequency at which the lists are recomputed. For example, the transport entity at the producer may recompute lists when a new object is added or 10 minutes after the last recomputation, whichever is *longer*. This imposes a minimum list update period and hence bounds overhead.

To prevent inconsistent list versions from jeopardizing delivery order to a consumer, a limited number of past versions may be maintained at the producer's transport entity. The transport entity at the consumer notes the version number of the first list it receives, $root/list/1$ and, in the rest of the data transfer, requests lists of the same version number.

Having said the above, we should note that InfoMax envisions data that does not change often. For example, once sensor data objects are collected, they are stored for a long period of time without modification. Similarly, once users issue tweets or images, they live in the blogosphere without modification until obsolete. The most common case of updates is to add new objects or new branches to the tree. We recommend that, in applications featuring frequent updates, the first branch under the consumer's prefix, $root$ should refer to a time window. Hence, each window, a new tree is constructed from objects in that window. That tree then remains fixed for the rest of time, once a new window starts. A consumer can request specifically the window(s) that they need under $root$.

Note that, name space design, is in general an important application-level issue for applications that serve data over information-centric networks. Application designers must consider other design implications as well, such as security and ease of routing. Those implications may be orthogonal to the naming hint suggested above, as they tend to affect the composition of the prefix $root$ itself, more so than the topology of the descending tree under it.

Fig. 1 depicts an example flow of the InfoMax protocol. In step 1, an application offers a new name space by creating an InfoMax producer. In step 2, a consumer is created who would like to receive a sampling of this content. In step 3, the consumer's transport entity sends an NDN interest in the first list object (called a "diversity interest" in the figure). The interest is routed by NDN to whoever has this list. Unless the content is already cached somewhere, the NDN interest packet makes it to the producer. On receiving this packet at the producer, the NDN layer invokes a callback to the

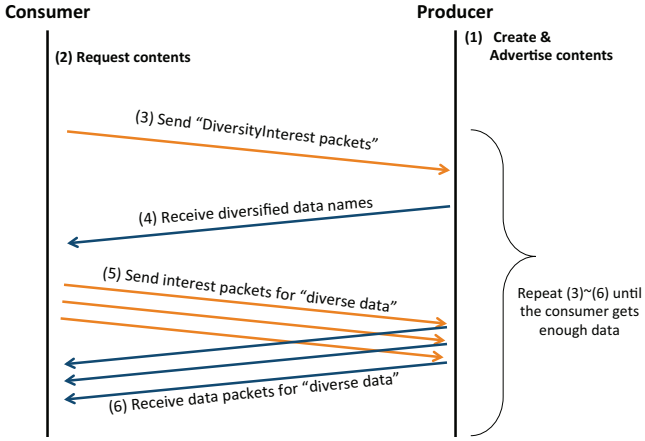


Fig. 1: Communication flow of InfoMax

InfoMax transport entity. The entity sends back the first list with the names of data objects that the consumer should ask for (step 4). On receiving this list, the consumer requests the data objects (step 5). These requests are fulfilled by whoever has the content. In the figure, the content is fulfilled again by the producer (step 6). The consumer may request more content, repeating steps 3-6 until satisfied or until all content under the tree is collected.

D. An Approximate Transmission Ordering Algorithm

An efficient implementation of the shortest-shared-prefix-first delivery algorithm is needed to reduce InfoMax overhead. In prior work, the authors proposed an optimal algorithm for diversifying data objects based on name similarity [4]. However, this algorithm is relatively slow. Instead, we developed a simplified algorithm that works as shown below.

```
simplePrioritizer(Node root, Node[] list)
while unvisited leaf nodes in tree do
    list.insert(getNextNode(root));
end

Node getNextNode(Node currentNode)
currentNode.count += 1;
if currentNode == leafNode then
    return currentNode;
end
for child = currentNode.children do
    if child has the smallest count among siblings with
    unvisited leaf nodes then
        getNextNode(child);
    end
end
```

Algorithm 1: Approximate least-shared-prefix algorithm

The algorithm starts with all branch counts initialized to zero, then traverses the tree from the root node taking, at each level of the tree, the least visited branch and incrementing its count. If there is a tie, the leftmost branch (of those involved in the tie) is taken. Once a leaf node is reached, the object at the leaf is returned and its name appended to the output list. The traversal starts again from the root for each subsequent object. The algorithm terminates when all leaves have been visited. The list generated in this process approximates shortest-shared-prefix-first. This is because an object that has the shortest shared prefix with previously listed ones must descend from some branch, B, not yet traversed, whose sibling have been traversed already. The count of that branch would therefore be zero. When the algorithm reaches the parent node at branch, B, say, node P, it will take branch B because it has the smallest count (specifically, its count is zero). Hence, the algorithm will correctly choose the branch leading to a shortest-shared-prefix object. The problem here is that the greedy nature of the algorithm might cause it to take a different branch earlier on, hence never reaching node P. This occurs when there is a tie in the counts somewhere closer to the root of the tree, causing the algorithm to descend into a different subtree (i.e., away from P). We investigate the efficacy of this algorithm in the evaluation section.

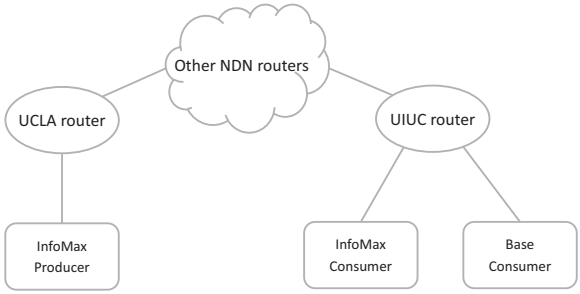


Fig. 2: Topology for measuring end-to-end delay

III. Evaluation

In this section, we evaluate InfoMax performance. This includes evaluating the overhead, investigating the impact of network caching, and assessing the degree to which the algorithm approaches the optimal shortest-shared-prefix-first transmission order. The experiments were done on a nation-wide testbed implementing the NDN protocol. There are 17 institutions participating in the NDN testbed in the world. The testbed map can be found in the NDN site [9]. These nodes are created for research purposes. Hosts can be connected to NDN routers by configuring and executing an NDN forwarding Daemon [10]. We used the above testbed for following experiments.

A. Transmission overhead

In order to measure end-to-end delay between a producer and a consumer, we created a producer in California (connecting to the UCLA NDN router) and a consumer in the Midwest (connecting the UIUC NDN router) as shown in Fig. 2.

To evaluate the extra overhead of InfoMax, we then created an additional consumer (also at UIUC) that is "clairvoyant" in that it knows exactly the topology of the producer's data

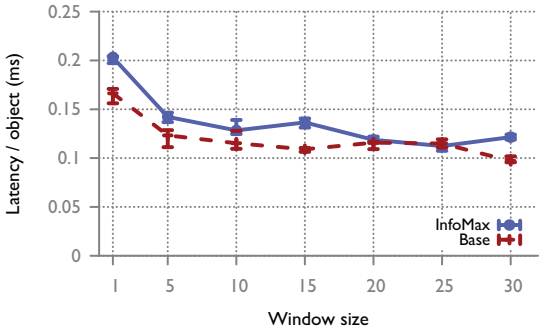


Fig. 3: Amortized per-object latency for different window sizes

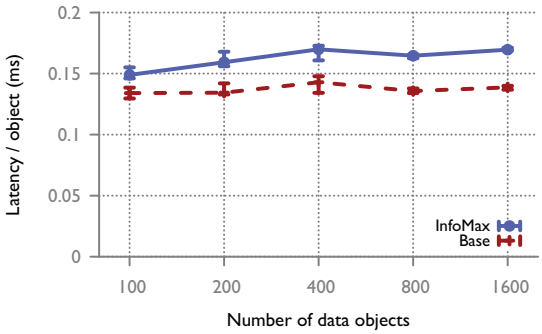


Fig. 4: Amortized per-object end-to-end delay as a function of the number of objects

tree. This consumer therefore simply requests data objects one a time by name via NDN, bypassing the InfoMax transport. This clairvoyant consumer is thus expected to show the best achievable performance in terms of latency and throughput.

Before comparing end-to-end latency of data retrieval in the two cases, we perform an experiment to decide the proper NDN pipelining window size. As mentioned in the design section, the window size in NDN is the number of interest packets that clients can send concurrently before receiving data packets matching these interests.

Fig. 3 shows end-to-end communication latency between a producer and a consumer for different window sizes. Both, InfoMax and the Base (clairvoyant) consumer, request 100 data objects from the InfoMax producer. The InfoMax consumer must of course also fetch the list object with data names. In the figure, the X-axis represents the consumer's window size and the Y-axis represents latency per data object. Note that, end-to-end latency decreases with increasing window size. This trend is opposed by packet drops (not shown) that increase. We set window size to 10 in the remaining experiments, offering reasonable latency and very rare packet drops.

With window size fixed as determined above, we perform the same experiment while varying the number of requested data objects. In Fig. 4, the X-axis denotes the number of data objects requested and the Y-axis denotes latency per one object. Considering that InfoMax exchanges additional packets and parses the list packets for data object names, we expect the

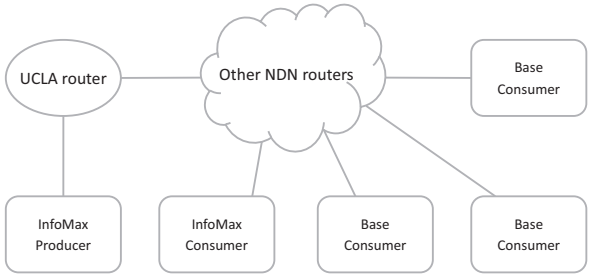


Fig. 5: Topology for testing NDN caching with multiple consumers

base (clairvoyant) client to outperform InfoMax. However, the added latency is not significant. Moreover, this extra latency could mitigated by successful NDN caching as will be shown in the following experiment.

B. Scaling delivery

A key benefit of NDN is in-network caching. InfoMax is carefully designed to leverage the NDN router cache. To demonstrate, we measure transmission rate at the producer side while increasing the number of InfoMax consumers. If NDN caching works well, the load on the consumer should not change. It should remain the same as with only one consumer. Fig. 5 shows the network topology used for this experiment. As before, the InfoMax producer is in California (connected to the UCLA NDN router). Four consumers are logically connected to different NDN routers across the testbed (University of Memphis, University of Michigan, Colorado State University, and UIUC).

Each consumer is programmed to request 100 data objects, repeatedly. We compare what happens when cache is enabled and disabled, respectively. The resulting transmission rate of the producer is shown in Fig. 6. In the figure, the X-axis and the Y-axis represent the number of consumers and the total transmission rate at the producer, respectively. Dotted lines in Fig. 6 show the producer's transmission rate in the presence of caching. Note that, it remains almost same as the number of consumers increases. This is because of successful NDN caching. Next, we disabled the cache function by setting cache time-to-live to be very short. As can be seen from the solid lines in Fig. 6, the resulting transmission rate of the producer increases almost linearly with the number of consumers in this case. The same trends are observed for both InfoMax and the base client. The experiments show that InfoMax does not increase the producer's transmission bandwidth needs.

C. Shortest-shared-prefix-first Ordering

InfoMax provides an approximate implementation of the shortest-shared-prefix-first order that approaches optimal behavior with low overhead. In order to show it, we first devise an experiment to measure how close InfoMax comes to an optimal shortest-shared-prefix-first algorithm. The optimal algorithm was described in the author's prior work and is used as the comparison point [4]. For this experiment, a random

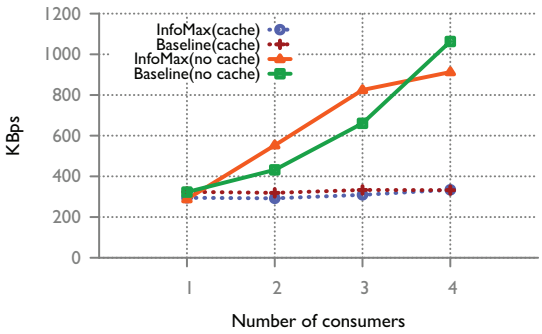


Fig. 6: Transmission rate of a producer with multiple consumers

tree generator is developed to create a diverse set of trees by controlling the total number of nodes, N, and the maximum number of children, C, under one parent. Different trees are created (with $N = 1600$ and $C = 20$), and the output of an optimal implementation of shortest-shared-prefix-first is compared to the approximate Infomax algorithm. Fig. 7 shows the result of this comparison, where the X-axis denotes the number of objects transmitted, and the Y-axis denotes the average number of shared segments (tree levels) between the current object name and previously sent data.

As can be seen in the graph, the number of shared segments in the name prefix increases monotonically as more objects are transmitted, indicating that diverse objects are transmitted first (i.e., those with a smaller prefix overlap) followed by progressively more redundant objects (with a higher prefix overlap). In application terms, the degree of redundancy increases as more objects are sent. Note also that the InfoMax curve is very close to the optimal, whereas the regular NDN curve (which, for the purposes of this experiment, corresponds to a leftmost child first traversal) is significantly higher meaning that more redundant objects are sent earlier, thereby reducing transmitted information value.

We perform the same experiments with two different sets of trees created by our generator. Both sets have the same number of nodes in a tree (N=1600), but different maximum number of children (C=5, 100). Trees having smaller C are deeper and those having bigger C are wider and shallower. Fig. 8 shows that deeper trees gain more from our algorithm in that the separation between the InfoMax and NDN curves is larger. This is because leaf nodes in a deeper tree have longer names, so the impact of a shortest-shared-prefix-first strategy is more pronounced. In the limit, if the tree depth is equal to 1, there would be no difference among the algorithms.

Although InfoMax is slightly worse than optimal (presented earlier [4]), it has a large computational advantage as can be seen in Fig. 9. The figure shows a large advantage for the approximate algorithm in terms of computation time.

IV. Application Examples

InfoMax, as previously discussed, relieves the consumer's application from having to figure out how best to sample a producer's abundant data. It also relieves the producer from

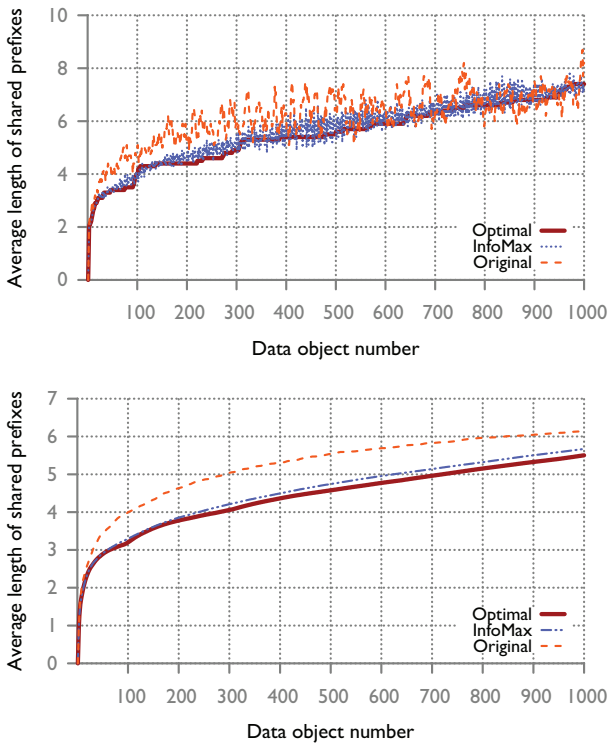


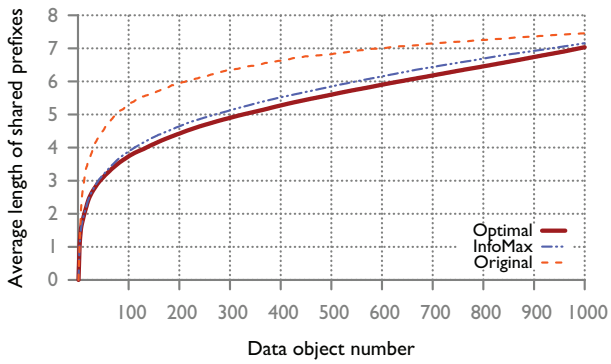
Fig. 7: Average length of shared prefix (in number of shared name segments) as a function of objects transmitted. Raw data (up) and smoothed data (down).

having to understand consumers' data summarization needs. The producer is simply responsible for naming and organizing its data objects, such that object categories with less similarity branch out earlier in the name space. Everything else follows naturally, as illustrated in the examples below.

A. Visual Tourism

As a simple example, we created a tourism application. It allows consumers to browse pictures of points of interest at destinations of choice. The pictures are organized hierarchically first by destination (e.g., "Rome", "Paris"), then by category of points of interest (e.g., "Ancient Landmarks", "Renaissance Landmarks", "Modern Landmarks", and "Religious Landmarks"), then by landmarks that belong to each category, and finally pictures of each landmark.

Suppose a consumer, a tourist, queries for ideas for a possible itinerary in Rome. The client software issues an InfoMax request to download pictures under the "/Tourism/Rome" prefix. Fig. 10 shows the first 6 images retrieved. We observe that the pictures are spread across various branches in the content tree, including ancient, middle age, and religious landmarks. They give a diverse view of Rome overall. The number of pictures in the overall data set exceeds 3000, with over 150 pictures of each single landmark. A random retrieval of these pictures would likely yield much redundancy. Instead,



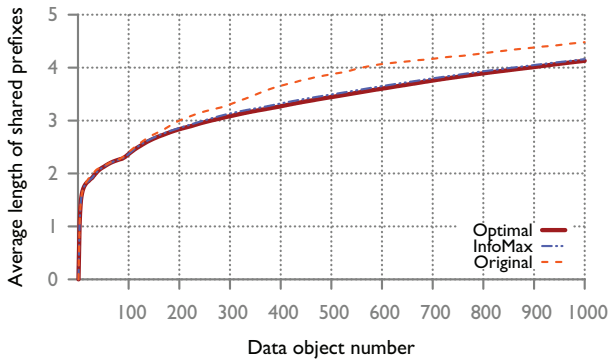


Fig. 8: Average length of shared prefix (in number of shared name segments) as a function of objects transmitted for deep trees (up) and wide trees (down).

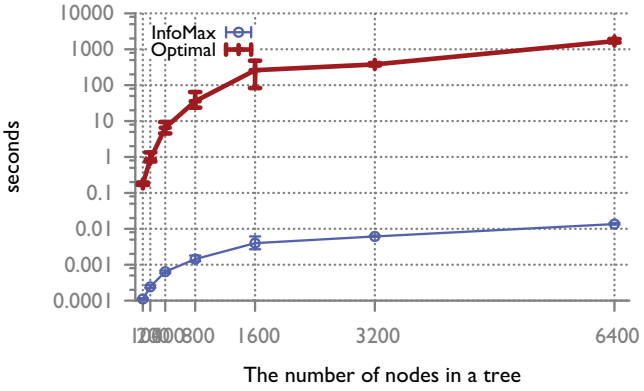


Fig. 9: Computation time for each prioritization algorithm

by following diverse top level branches first, the shortest-shared-prefix-first algorithm automatically retrieved a diverse set of pictures. No application semantics were needed to achieve this diversification.

Next, the consumer wishes to take a closer look at Rome's ancient landmarks. The client software queries InfoMax for "/Tourism/Rome/Ancient". The results are shown in Fig. 11 Again, they come from different branches at this level of the

Fig. 10: Top-level query – overview of Rome

Fig. 11: Second-level query – scenes from ancient Rome

tree, showing different ancient landmarks. Finally, the consumer wishes to see the Colosseum. The software queries InfoMax for "Tourism/Rome/Ancient/Colosseum". At this level only Colosseum pictures are retrieved as shown in Fig. 12.

Two points are notable in this example. First, the retrieved pictures, in each case, constituted a sampling of a data set that contained *orders of magnitude more matching entries* to the consumer's specific query, most of which were partially redundant. Among these *matching* entries, an appropriately diverse small sampling was returned. Second, such diversification of retrieved information occurred with no use of application-specific information. It was simply an artifact of following the generic shortest-shared-prefix-first retrieval order by the InfoMax transport entity.

B. Twitter Search

Next, we describe a Twitter-based news feed application utilizing InfoMax transport [11]. In this application, our pro-

Fig. 12: Query on the Colosseum

TABLE III: Top-level Kirkuk News Feed

S1	RT @Mojahedineng: #PMOI #Iran Iran: Three Kurdish teachers arrested to prevent teachers protest gathering: http://t.co/DQnqfniFyQ
S2	RT @DerekStoffelCBC: Lets remember journalist Kenji Goto as a brave, humane man, not an #ISIS victim. http://t.co/Ih4Teuz33e #CBC
S3	RT @MEAIndia: Beginning the day with good news. 11 Indian nurses from Kerala, evacuated from Kirkuk, to return home from Erbil on 7 February
S4	RT @peopIeofearth: A Kurdish fighter walks with his child in the streets of Kobani, Syria after they recaptured it from ISIS militants.
S5	RT @Alladin_Al: Y didnt any news Chanel broadcast the muslims being burned in Burma? Or are they only focused on their beloved "ISIS"
S6	RT @grasswire: A Kurdish marksman stands atop a building as he looks at the destroyed Syrian town of #Kobane. Photo @Kilicbil #AFP
S7	RT @salar_dd: #Turkish special forces member: We have launched attacks on #Kurdish villages; killed babies then blamed #PKK on TV
S8	RT @itirerhart: @BBCr4today: Kurdish victory in Kobane: Islamic State militants have been driven out of the Syrian town.
S9	RT @_Kurda_: #Peshmerga Destroyed Huge Islamic Jihadist Armor in #Kirkuk by MIlan Rockets #TweitterKurds #Kobane #JordanianPilot
S10	RT @TheDarlingBeast: 1988, #Kurdish children lined up to be slaughtered under Iraqi regime. Look at that innocent smile, so unaware.

TABLE IV: Kirkuk: More on Kobane Attack

S8	RT @itirerhart: @BBCr4today: Kurdish victory in Kobane: Islamic State militants have been driven out of the Syrian town.
D1	VIDEO: Awaiting return to war-torn Kobane http://t.co/3YeFlCPvvv Days after Kurdish forces drove Islamic State militants from Kobane
D2	Kurdish forces engage in sporadic battles with Islamic State militants around the Syrian town of Kobane, as they fight to expel IS
D3	Kurdish forces have driven Islamic State (IS) militants from Kobane, activists say, ending a four-month battle... http://t.co/4RJVgha2jA
D4	SYRIAN Kurdish fighters have seized dozens of villages from Islamic State jihadists around the town of Kobane. http://t.co/7oLYiwlKC2
D5	Iraqi Kurdish forces retook an oil station Saturday that had been seized by Islamic state militants, but there... http://t.co/oM5UjkuYv8
D6	"Earlier gains were fueled by reports of Islamic State militants striking at Kurdish forces" talking oil prices http://t.co/GsCQU1pDol
D7	The Kurds Kurdish forces engaged in sporadic battles with Islamic State jihadists around the Syrian town of Kobane on Saturday
D8	Islamic State moves into south-west of Syrian Kurdish town - The Malaysian Insider (themalaysianinsider) http://t.co/3El3TuiO8m
D9	RT @indianews: After Kobani, Islamic State may target Kurdish town of Afrin next: Western and Arab powers that ...
D10	[Xinhua News] IS militants flee surrounding of Syrian Kurdish city: activists: The militants of the Islamic St... http://t.co/ep5ebx1ghb

TABLE V: Kirkuk: More on Kenji Goto

S2	RT @DerekStoffelCBC: Lets remember journalist Kenji Goto as a brave, humane man, not an #ISIS victim. http://t.co/Ih4Teuz33e #CBC
D1	My condolences RT @DerekStoffelCBC Lets rmmbr Kenji Goto as brave man, not #ISIS victim. http://t.co/NIuoBLHEve #CBC http://t.co/qjJzoWWTG1
D2	ISIS exhibited the video by which Kenji Goto is killed. http://t.co/6KahTbaFYr http://t.co/oiutIpdCIi
D3	Kenji Goto's wife announced the declaration which requests to free him in ISIS. http://t.co/FY0Z1oKUbg http://t.co/LDmLjJfkMS
D4	Free Kenji Goto. #KenjiGoto #ISIS
D5	kenji goto death official execution by isis: http://t.co/DW5Gt72x5w @YouTube
D6	RT @Acosta: POTUS statement on ISIS killing of Kenji Goto http://t.co/IWAtUE7ypw
D7	Caroleina2 @NewsHour Kenji #Goto R.I.P. #ISIS are depraved bloodthirsty maniacs. Not part of humanity.
D8	Rest in peace Kenji Goto. ISIS does not represent Islam, period.
D9	RT @Yunghi: "Kenji Goto's reporting is voice of humanity in times of atrocity" https://t.co/hFKdisfRDt #kenjigoto #isis
D10	RT @rcampbelltokyo: More samples from Kenji Goto's blog! #IAMKENJI #ISIS #JapaneseHostage. https://t.co/CXsxCCl3xX

ducer crawls Twitter for tweets on events that it wants to report a Twitter-based news-feed on. These tweets have various degrees of similarity. Some are plain retweets without new information, some are retweets with added information, some provide information related to an event from possibly different perspectives, and some are entirely different. As a result, these tweets can be hierarchically clustered by similarity as described by a subset of the authors in prior work [5]. The resulting hierarchical structure is used to generate a content tree in the NDN name space. This content tree is served by the InfoMax producer, offering news about ongoing events at a configurable level of granularity to consumers.

Specifically, we collected tweets on recent ISIS attacks in the Iraqi Kurdish town of Kirkuk. Around 200K tweets were collected (from Jan 31 to Feb 11, 2015) matching the keywords "ISIS", "Kirkuk", and "Kurdish" or the geographic locations around Kirkuk, Iraq. To prioritize what may be important for the consumer, siblings sharing a common prefix were ordered left to right in a decreasing order of the number of their descendants. (This takes advantage of the generic InfoMax tie breaking rule, described earlier, that takes the leftmost branch of those involved in a tie.)

Consider a consumer who specifies that they want a 10-tweet summary of the aforementioned news feed. Following the InfoMax shortest-shared-prefix-first order, the producer serves content from ten different top-level branches of the content tree. The resulting output is shown in Table III. (Only English tweets are selected for readability in this example.)

After reading the result, the consumer is interested in the details of a particular event. The client software allows it makes a followup query to retrieve more results on a given tweet (similar to retrieving results on the Colosseum in the tourism example). This is implemented by having the software ask InfoMax for the subtree of this tweet's parent. (Note that, the software knows the name of the parent node because it already retrieved the tweet in question from the producer, including its full name in the producer's name space). Table IV, and Table V show 10 tweets on topics $S8$ and $S2$ from Table III, respectively. Observe that these subsequent queries return much more targetted results.

The point to emphasize here is that we were able to control the level of specificity of query results simply by following the shortest-shared-prefix-first transmission order (starting with different nodes in the tree). This order was efficiently approximated in the InfoMax transport layer that is completely unaware of application semantics. The only interface each application used in the above examples was to request content from subtrees, matching a specific prefix in the producer's name space. Very importantly, not all content matching the consumer's query was retrieved but only a very small sampling in each case. The size of the sample was determined by the consumer. By following the InfoMax retrieval order, the

samples were properly diversified to offer a broad coverage of each query using the small number of retrieved objects. We believe this diversified sub-sampling capability will be of increasing value to many emerging data-centric applications, where data over-abundance is the norm.

V. Related work

With the growing demand on data oriented applications, many recent proposals advertised moving from host- to content-oriented networking. Information-centric networks, such as NDN, espouse named content rather than named hosts, as a central abstraction. The result retains the simplicity and scalability of IP but offers better security, delivery efficiency, and disruption tolerance.

Our paper leverages NDN [12] to offer a transport layer solution that reduces redundancy in a representative category of modern data sets. The problem of data redundancy was described initially by a subset of the authors in Photonet [13], [14]. A protocol was designed to prioritize images forwarding and replacement in a disruption-tolerant network depending on the degree of similarity (or dissimilarity) among them. Minerva [15] presented an information-centric programming paradigm and toolkit for social sensing that generalizes PhotoNet to arbitrary content types. Related ideas that exploit a hierarchical data organization were presented in recent work [16], [17].

Our paper falls in the category of saving resource by reducing data transmission volume. Much work was done in this area over the decades, from compressing data [18], [19], [20] to selectively sending a subset [21], [22]. Our research is closer to the latter category. It is unique in exploiting a generic scheme that sits in a transport layer and does not require application-specific knowledge. An advantage of this approach is that different applications can use it, instead of relying on their own application-specific solutions.

Work that comes most closely to ours is the Information Funnel [4]. It focuses on the data collection side, whereas the current paper focused on dissemination. The two combined offer an end-to-end solution for data-intensive services that collect data from many sources and offer it to many consumers in an information rich world marked by data over-abundance.

VI. Conclusions and Future Work

In this paper, we proposed a new transport layer protocol to sub-samples big data sets. This protocol is built on top of NDN and exploit its hierarchical name structure. By observing that data objects with a longer shared prefix have more semantic similarity, InfoMax produces (an approximation of) the shortest shared prefix order among transmitted data objects to minimize data redundancy. It allows multiple consumers to retrieve data at different configurable levels of detail from one producer while maximally leveraging NDN caching to minimize producer overhead. InfoMax has been deployed on a nation-wide testbed. Analysis of long-term experiences with this deployment is deferred to a subsequent publication.

References

[1] E. Elijah, "Information output is growing exponentially: 3 strategies for laboratory data management," http://accelrys.com/, May 2014.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Computer Communication Review (CCR)*, July 2014.

[3] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 4, pp. 117–130, Aug. 1996. [Online]. Available: http://doi.acm.org/10.1145/248157.248168

[4] S. Wang, T. Abdelzaher, S. Gajendran, A. Herga, S. Kulkarni, S. Li, H. Liu, C. Suresh, A. Sreenath, H. Wang, W. Dron, A. Leung, R. Govindan, and J. Hancock, "The information funnel: Exploiting named data for information-maximizing data collection," in *Proc. International Conference on Distributed Computing in Sensor Systems*, May 2014.

[5] M. T. Amin, S. Li, M. R. Rahman, P. Seetharamu, S. Wang, T. Abdelzaher, I. Gupta, M. Srivatsa, R. Ganti, R. Ahmed, and H. Le, "SocialTrove: A self-summarizing storage service for social sensing," in *Proc. International Conference on Autonomic Computing (ICAC'15)*, July 2015.

[6] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *Network, IEEE*, vol. 28, no. 3, pp. 50–56, May 2014.

[7] "C++ infomax," https://github.com/infomaxndn/InfoMax-CXX.

[8] "Infomax consumer for ios," https://github.com/infomaxndn/InfoMax-ConsumerSwift.

[9] Washington University in St. Louis, "NDN testbed," http://ndnmap.arl.wustl.edu.

[10] NFD developers, "NFD tutorial," http://named-data.net/doc/NFD/current/INSTALL.html.

[11] "Apollo twitter search ios client built on infomax," https://github.com/infomaxndn/ApolloExample.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," *SIGCOMM*, December 2009.

[13] M. Y. S. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang, "Photonet: A similarity-aware picture delivery service for situation awareness," *Proceedings of the 2011 IEEE 32nd Real-Time Systems Symposium*, pp. 317–326, November 2011.

[14] M. Y. S. Uddin, M. T. A. Amin, T. Abdelzaher, A. Iyengar, and R. Govindan, "Photonet+: Outlier-resilient coverage maximization in visual sensing applications," *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, April 2012.

[15] S. Wang, S. Hu, S. Li, H. Liu, M. Uddin, and T. Abdelzaher, "Minerva: Information-centric programming for social sensing," *Proceedings of the 22nd International Conference on Computer Communications and Networks*, pp. 1–9, July 2013.

[16] W. Dron, A. Leung, M. Uddin, S. Wang, T. Abdelzaher, R. Govindan, and J. Hancock, "Information-maximizing caching in ad hoc networks with named data networking," *Proceedings of Network Science Workshop*, pp. 90–93, 2013.

[17] S. Kumar, L. Shu, S. Gil, N. Ahmed, D. Katabi, and D. Rus, "Carspeak: A content-centric network for autonomous driving," *Proceedings of ACM SIGCOMM*, pp. 259–270, 2012.

[18] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer, "Network correlated data gathering with explicit communication: Np-completeness and algorithms," *IEEE/ACM Transactions on Networking*, pp. 41–54, 2006.

[19] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 24, 2008.

[20] M. C. Vuran, zgr B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks Journal (Elsevier)*, pp. 245–259, January 2004.

[21] H. Gupta, V. Navda, S. Das, and V.Chowdhary, "Efficient gathering of correlated data in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 4, January 2008.

[22] C. Liu, K. Wu, and J. Pei, "An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation," *IEEE Transaction on Parallel Distributed System*, pp. 1011–1023, 2007.