# DAPES: Named Data for Off-the-Grid File Sharing with Peer-to-Peer Interactions

# DAPES: Named Data for Off-the-Grid File Sharing with Peer-to-Peer Interactions

Spyridon Mastorakis
University of Nebraska, Omaha
smastorakis@unomaha.edu

Tianxiang Li
UCLA
tianxiang@cs.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

***Abstract*—This paper introduces DAta-centric Peer-to-peer filE Sharing (DAPES), a data sharing protocol for scenarios with intermittent connectivity and user mobility. DAPES provides a set of semantically meaningful hierarchical naming abstractions that facilitate the exchange of file collections via local connectivity. This enables peers to "make the most" out of the limited connection time with other peers by maximizing the utility of individual transmissions to provide data missing by most connected peers. DAPES runs on top of Named-Data Networking (NDN) and extends NDN's data-centric network layer abstractions to achieve communication over multiple wireless hops through an adaptive hop-by-hop forwarding/suppression mechanism. We have evaluated DAPES through real-world experiments in an outdoor campus setting and extensive simulations. Our results demonstrate that DAPES achieves 50-71% lower overheads and 15-33% lower file sharing delays compared to file sharing solutions that rely on IP-based mobile ad-hoc routing.**

***Index Terms*—Data distribution, Off-the-grid file sharing, Named Data Networking**

## I. Introduction

"Off-the-grid" communication includes scenarios, where Internet connectivity may not be available, since the backbone infrastructure may be damaged (e.g., disaster recovery) or absent (e.g., battlefield, rural areas). Data sharing in such scenarios is vital for the dissemination of critical information (e.g. damage status for disaster recovery) and needs to be done through local network connectivity among the communicating entities. The communicating entities may also be mobile with intermittent connectivity to each other and the network topology dynamic, introducing new challenges to data sharing.

Although the communicating entities are inherently interested in the data to share, existing solutions that run on top of the IP-based network architecture [17], [35] typically rely on Mobile Ad-hoc Networking (MANET) routing protocols, such as DSDV [34] and AODV [33], to establish reachability to the IP address of each entity. After that, the actual data delivery can begin. Moreover, in off-the-grid scenarios, IP address configuration becomes a challenge; a number of existing solutions have been proposed [32], [25], [28], which share the goal of assigning an IP address to each entity that does not collide with others. That is, in the context of off-the-grid communication, IP addresses are merely unique node identifiers, since the node location may constantly change.

In this paper, we argue that a data-centric approach to off-the-grid file sharing aligns with the objective of the communicating entities, namely the inherent interest in the data they would like to share. In line with this assertion, we propose DAta-centric Peer-to-peer filE Sharing (DAPES), which defines semantically meaningful hierarchical naming abstractions that identify the shared data directly. These names are independent of the location of the entity that produced the data or the underlying connectivity. Through these semantically meaningful names, DAPES also conveys compactly encoded information about the data that the participants of the file sharing process, called *peers*, have and facilitates transmission prioritization among peers for efficient data sharing.

DAPES runs on top of Named Data Networking (NDN) [42], which provides a request/response communication model, directly utilizing the names defined by DAPES. DAPES leverages NDN's cryptographic primitives that bind the content of each network layer packet to its name, enabling peers to reason about data provenance and integrity. DAPES extends the NDN data-centric network layer abstractions to make use of any and all the means of connectivity, being able to fetch data from any peer that can provide it in the network. As a result, a "traditional" MANET routing protocol for communication across multiple wireless hops is no longer needed. Expressing the DAPES operations through semantically meaningful names, used directly by the underlying network, enables peers to make forwarding decisions based on what data is available through multiple hops over time.

The contributions of our work are the following:

- We propose and design DAPES, a data-centric protocol for peer-to-peer file sharing in off-the-grid communication scenarios. DAPES offers unified mechanisms to maximize the utility of transmissions and mitigate collisions due to simultaneous transmissions (Section IV). As a result, peers "make the most" out of each (short-lived) encounter with others, minimizing the number of required transmissions. DAPES also extends NDN's data-centric forwarding plane to build short-lived knowledge about the data available around peers. In this way, multi-hop communication is achieved through an adaptive hop-by-hop forwarding/suppression mechanism (Section V).
- We implement a DAPES prototype, which we evaluate through real-world experiments in an outdoor campus setting and extensive simulations (Section VI). Our results demonstrate that DAPES achieves 50-71% lower overheads and 15-33% lower file sharing delays than IP-based solutions that rely on MANET routing.

To the best of our knowledge, DAPES is one of the very first efforts to offer a concrete design and implementation of a data sharing solution in dynamic off-the-grid setups on top of a data-centric network substrate.

## II. Background & Prior Work

In this section, we present an overview of the NDN architecture, prior related work to discuss how DAPES is inspired, but also differs from prior efforts, and a sample use-case that we use to elaborate on the DAPES design throughout the paper.

### *A. NDN Overview*

In NDN, each data packet is assigned a unique name at the time of its production. This name is used as the data identifier by the network layer and is independent of the underlying network connectivity. The NDN communication paradigm is receiver-driven; *data consumers* send requests, called Interest packets, for named data packets generated by *data producers*. Data names are semantically meaningful, hierarchically structured and can contain a variable number of components. For example, a consumer sends an Interest with a name "`/cnn/daily-news/headlines`" to fetch the headlines of the daily news from CNN. NDN builds communication security directly into the network architecture, since data producers cryptographically sign each data packet at the time of generation. The signature binds the content of a data packet to its name, so that a consumer can authenticate the data directly using the producer's public key [1].

NDN Forwarding Daemons (NFDs) [2] can cache received data packets to satisfy future requests for the same data, given that each data packet is named and secured directly at the network layer. When an NFD receives an Interest, it first checks whether the requested data exists in its local Content Store (CS), as illustrated in Figure 1. If no cached data is found, the Interest is checked against the entries of the Pending Interest Table (PIT), where state is maintained about the Interests that have been forwarded, but the corresponding data has not been received yet. If a pending Interest with the same name exists in PIT, no further forwarding is performed, since data is expected to be received. If no matching Interest is found, NFD determines how to forward the Interest based on a Longest Prefix Match (LPM) between the Interest name and the entries in its Forwarding Information Base (FIB). A data packet uses the state in PIT, created by the corresponding Interest at each-hop NFD, to follow the reverse path back to the requesting consumer(s).

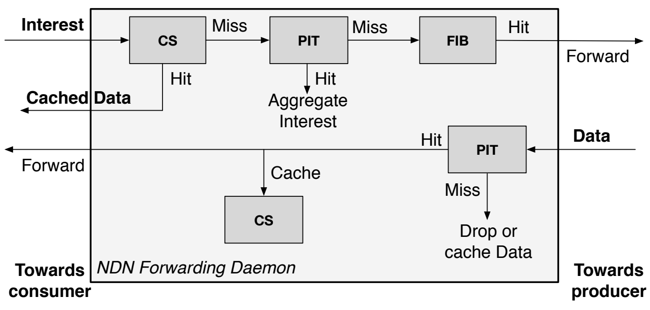


Fig. 1: Packet processing by NFD

### *B. Prior Related Work*

**Prior work in IP:** BitTorrent [6] is the most popular peer-to-peer file sharing application, focusing on infrastructure IP-based networks. Efforts to adapt BitTorrent to work in MANET [36], [12], [37], [17] largely rely on MANET routing [13], [15], [33] for path discovery and maintenance between peers. Mobility introduces additional challenges, since established paths break and new ones have to be established. Previous work [30], [18], [16], [35] has advocated that alternative solutions for multi-hop communication need to be explored (e.g., application-layer gossiping, network coding, link-layer flooding). Security is not considered in the routing and the data sharing process [41], while IP address configuration in such infrastructure-free environments is a challenge on its own [32], [25], [28].

**Prior work in NDN:** nTorrent [23], inspired by BitTorrent, is an NDN-native application for peer-to-peer file sharing in infrastructure networks. FileSync/NDN [21] implemented a file synchronization system in a shared directory among peers, while Detti et al [8], [9] and Malabocchia et al [22] studied the design of peer-to-peer video streaming applications. Previous work has also studied the design of general purpose architectures and forwarding mechanisms [3] for MANET in NDN. E-CHANET [4] is such an architecture that runs on top of IEEE 802.11 and focuses on providing stable paths and reliable transport functions towards named data. Varvello et al. [38] and Meisel et al. [26] performed an initial exploration of the design space for various MANET challenges in NDN, such as resource discovery and multi-hop forwarding. Finally, Li et al. [20] designed and implemented DDSN, a protocol for distributed dataset synchronization under disruptive network conditions.

**How DAPES is inspired and how it differs from prior work:** DAPES builds on prior work on peer-to-peer file sharing. BitTorrent uses a torrent-file that contains metadata about the shared file collection (e.g., tracker IP address, cryptographic hash of data for integrity verification), helping peers initialize their file downloading process. In a similar manner, nTorrent uses a metadata file that contains the names and the hashes of the data to request. DAPES, inspired by BitTorrent and nTorrent, uses cryptographically signed metadata (Section IV-C) to help peers learn the names of the data to request and verify its integrity. BitTorrent peers use a bitmap to keep track of the data they have and leverage the "Rarest Piece First" (RPF) strategy to replicate data. DAPES peers also use a bitmap to encode the data peers have in a compressed manner. We explore different ways for DAPES peers to advertise this information in order to increase the efficiency of the data sharing process under intermittent connectivity (Section IV-D). We also propose variations of the RPF strategy, which are specifically designed to maximize the replication of rare data in dynamic communication scenarios (Section IV-E).

Solutions for distributed dataset synchronization, such as DDSN [20], focus on the exchange of dynamic content, contrary to DAPES that focuses on the exchange of static content among peers. Preliminary design space explorations [26], [38] did not result in concrete protocol designs and solutions, while frameworks such as E-CHANET [4] did not fully exploit the data-centricity of the underlying NDN architecture to achieve their goals.

DAPES is a concrete data-centric framework for peer-to-peer file sharing in off-the-grid scenarios. DAPES functions are achieved through a set of mechanisms that maximize the utility of each single transmission (Sections IV-D and IV-F). At the same time, DAPES mitigates collisions due to simultaneous peer transmissions, facilitating file sharing in cases of encounters among multiple peers (Section IV-F). Thanks to all these mechanisms, peers can "make the most" out of each (potentially short-lived) encounter with others, maximizing the efficiency of the file sharing process. Furthermore, DAPES extends the underlying data-centric forwarding plane to identify what data is available over multiple wireless hops, thus making accurate forwarding/suppression decisions (Section V).

### C. Example Use-Case

Our example use-case (Figure 2) assumes a rural area, where residents would like to share with other residents information about damaged parts of the infrastructure (e.g., a damaged bridge). This information needs to be resiliently, securely, and efficiently shared with as few transmissions as possible (i.e., minimal overhead and energy consumption), given that the resident devices may have limited battery power (e.g., mobile phones, tablets). We assume that residents have an instance of the DAPES application running on their device, which names individual files, segments a file into network-layer data packets and signs these packets[1], groups individual files together to create *a collection of files*, and shares files with others.

Let us assume that a resident takes a picture of a damaged bridge and, through the DAPES application, names this picture (file) as "`bridge-picture`". The resident also creates another file that contains information about the bridge location (e.g., longitude, latitude, description of surroundings) and names this file as "`bridge-location`". Finally, the resident, through the DAPES application, creates a file collection consisting of these two files with a name "`/damaged-bridge-1533783192`" that includes the unix-timestamp of when the collection was generated. Each file in the collection consists of a number of individual data packets signed by the private key of the resident, who acts as the collection producer and starts disseminating the file collection data, aiming to notify other residents about the damage.

The data is disseminated through: (i) peer-to-peer interactions among residents as they move around in the rural area (e.g., in Figure 2, resident A encounters residents B, C, and D as A moves around in the rural area, while resident E encounters F), and (ii) (stationary) data repositories ("repos" for short) locally deployed (e.g., in a rest area) to enhance data availability through collecting and serving data from/to residents [29] (e.g., in Figure 2, resident G is at a rest area, where a repo has been deployed).

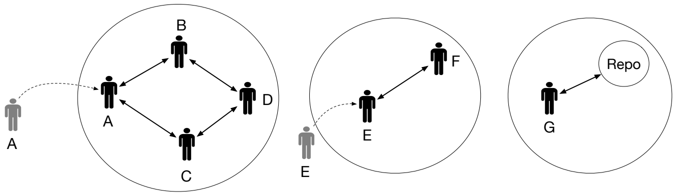


Fig. 2: Example of an off-the-grid communication scenario

## III. DAPES Design Overview

Our design aims to achieve efficient data sharing with low overhead among multiple peers under dynamic and adverse network conditions through: (i) a semantically meaningful and hierarchical namespace (Section IV-A) directly utilized by an underlying data-centric network substrate, (ii) secure initialization of the data sharing process through cryptographically signed metadata (Section IV-C), (iii) compact encoding of what collection data peers have (Section IV-D), and (iv) mechanisms for efficient data discovery, sharing, and collision mitigation (Sections IV-B, IV-E, and IV-F respectively).

Given that peers may constantly move, a mechanism is needed to discover when peers are within the communication range of each other and what file collections they have (step 1 in Figure 3). Peers need to securely initialize their data sharing process by: (i) authenticating that the file collection producer can be trusted, and (ii) learning the names of the data to request in order to retrieve a file collection and verifying the integrity of the retrieved data packets. To achieve that, the file collection producer generates and signs a metadata file for the collection. When peers discover for the first time a file collection of interest through an encountered peer, they retrieve and authenticate the collection metadata (step 2 in Figure 3). To verify the authenticity of others, including the producer of the file collection, we assume that peers have common "local" trust anchors (e.g., among the residents of the rural area) established [43]. Based on these common trust anchors, peers verify the metadata signature and decide whether they trust the collection producer.

To reduce bandwidth consumption and communication delay, peers exchange compactly encoded information about the data they have, called "data advertisements" (step 3 in Figure 3). They prioritize the retrieval of rare data in the context of off-the-grid communication (i.e., data missing by most peers around them) through variations of the basic RPF strategy (step 4 in Figure 3), and use a random timer for collection data transmissions to avoid collisions. To ensure that peers are aware of as many of the available packets as possible within their communication range, they make use of a prioritization scheme for data advertisement transmissions. For MAC layer communication, peers use IEEE 802.11 in ad-hoc mode under the same SSID and channel number [7].

**Communication over multiple wireless hops:** In addition to maximizing the data sharing benefits across a single wireless hop, DAPES achieves low overhead communication over multiple wireless hops. DAPES is able to make dynamic Interest forwarding/suppression decisions by assessing whether a forwarded Interest is likely to bring data back. To achieve that,

[1] We assume that each resident has a pair of public and private keys to sign the packets that it generates.

peers keep track of the available data around them. When peers speculate that forwarding a received Interest will not retrieve the requested data, they suppress the Interest, while they forward received Interests when they deem that the requested data may be available around them (Section V).

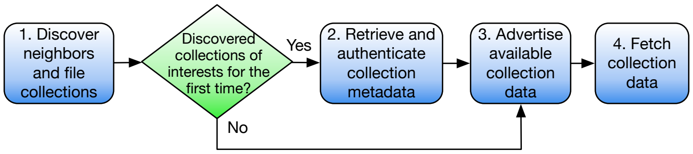

1. Discover neighbors and file collections
Discovered collections of interests for the first time?
Yes
No
2. Retrieve and authenticate collection metadata
3. Advertise available collection data
4. Fetch collection data

Fig. 3: DAPES design overview

## IV. DAPES Design Components

In this section, we present the components of the DAPES design in detail.

### A. Namespace Design

Our goal is to enable peers to easily identify specific data of interest in a data collection (e.g., peers might not be interested in all, but only specific files in a collection). Therefore, we design a semantically meaningful and hierarchical namespace that enables peers to identify the name of a file collection, an individual file in a collection, and a data packet in a file. We identify individual packets through a sequence number, which allows us to compactly encode information about which packets in each file peers have, as we explain in Section IV-D. In our sample use-case (Section II-C), the collection of files has a name "`/damaged-bridge-1533783192`", which includes a unix-timestamp that specifies when the collection was generated. The first packet in the first file (picture with name "`bridge-picture`") has a name "`/damaged-bridge-1533783192/bridge-picture/0`".

### B. Peer & File Collection Discovery

Given that peers are mobile and their position may constantly change, we need a mechanism to make them aware of when others are within their communication range and what collections they have. To achieve that, each peer periodically broadcasts signaling Interests, called *discovery Interests*. To mitigate the overhead caused by short periods of signaling Interests, peers dynamically adjust their transmission period. Peers broadcast signaling Interests more frequently when they have recently encountered others, while their transmissions become less frequent when they are in isolation from others. Peers, receiving a discovery Interest, send a *discovery data packet* back that contains the name of the metadata files for the file collections they have[2]. In this way, peers can discover the file collections that others can offer. The discovery namespace consists of the application name and the name component "discovery" ("`/dapes/discovery`").

This mechanism is implemented at the application layer. We aim to run on top of the existing IEEE 802.11 in ad-hoc mode, which incorporates a beaconing mechanism for clock synchronization and the discovery of neighbors, however, it does not provide any feedback to the upper layers of the network architecture [7]. In the example of Figure 2, each peer periodically broadcasts discovery Interests, which helps F and E discover each other and the collections they have after E moves within the communication range of F.

### C. Metadata For Secure Initialization

When peers discover for the first time a file collection of interest through an encountered peer, they need to securely initialize their data sharing process. To achieve that, peers retrieve and authenticate the metadata file, which consists of one or more data packets generated and signed by the collection producer. The metadata also helps peers to discover the data namespace (name of files and individual packets) and verify the integrity of each data packet in the file collection, without having to verify its signature, which would be computationally more expensive. In the rest of this section, we present alternative metadata encoding formats (Figure 4) and how each one achieves the above goals. These encodings involve a trade-off between the size of the metadata and how soon the integrity of each received packet can be verified.

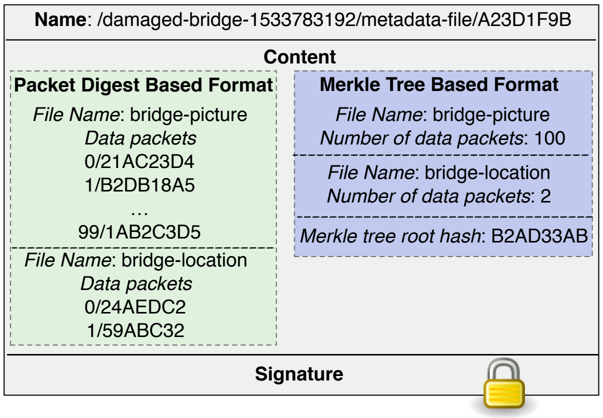
**Packet digest based format:** Based on our hierarchical namespace, all the files in a collection and the packets in an individual file share a common name prefix. As a result, the metadata file can contain a list of "subnames" in the form of "`[packet-index]/[packet-digest]`" for each individual file. The subname refers to the index of a specific packet in the file followed by the packet's digest. To construct each packet's name, a peer first appends each subname to the name of the corresponding file, and then appends the resulting name to the collection name. Each subname's digest enables peers to verify the integrity of a packet as soon as it is received. This approach can result in large metadata files that need to be segmented into multiple network layer packets. Given that peers might have limited connection time during an encounter, they might need multiple encounters to fetch the entire metadata file.

**Merkle tree based format:** Peers verify the integrity of the packets through a Merkle tree [27], whose root hash is generated by the collection producer and is included in the metadata content. There can also be multiple Merkle trees (e.g., one per individual file in the collection), thus, multiple root hashes can be included in the metadata. The metadata can typically fit into a single network layer packet, but all the packets in a tree need to be retrieved before their integrity can be verified. To construct each packet's name, peers first append the packet index (e.g., for a file with 100 packets, the index of the first packet is 0 and of the last packet is 99) to the name of the corresponding file, and then append the resulting name to the collection name.

### D. Data Advertisements

Given that encounters among peers might be short-lived, peers need to advertise what data they have for a collection in a compact way. Since data is hierarchically and sequentially named, we take advantage of a *bitmap* data structure. Each bit refers to an individual packet, having a value of 1 if the peer has this packet and 0 if this packet is missing. After a

[2] The sender of a discovery data packet can learn the name of the metadata files of peers in its neighborhood by sending its own discovery Interest.



**Name**: /damaged-bridge-1533783192/metadata-file/A23D1F9B

**Content**

**Packet Digest Based Format**

*File Name*: bridge-picture
*Data packets*
0/21AC23D4
1/B2DB18A5
...
99/1AB2C3D5

*File Name*: bridge-location
*Data packets*
0/24AEDC2
1/59ABC32

**Merkle Tree Based Format**

*File Name*: bridge-picture
*Number of data packets*: 100

*File Name*: bridge-location
*Number of data packets*: 2

*Merkle tree root hash*: B2AD33AB

**Signature**

Fig. 4: Metadata file format example

peer downloads the collection metadata, it creates a bitmap of 0s for this collection by ordering the data packets based on the relative position of the files in the metadata and the position of the packets in each file. For the metadata of Figure 4, the first bit of the bitmap refers to the first packet of the first file ("`bridge-picture`"), the second bit to the second packet of this file, etc. The first packet of the second file ("`bridge-location`") corresponds to the 101st bit of the bitmap, and the second to its 102nd bit.

After discovering the file collections an encountered peer has, peers send an Interest, called a *bitmap Interest*, for each collection they are interested in. Each such Interest carries the sender's bitmap for the corresponding collection. A peer receiving such an Interest sends its own bitmap back in the content of a *bitmap data packet*. In the example of Figure 2, peer E receives F's discovery data and sends a bitmap Interest carrying its bitmap. F receives E's bitmap Interest and sends back a bitmap data packet containing its own bitmap.

**Encounters among multiple peers:** When multiple peers meet each other (e.g., A moves into the communication range of B, C and D in Figure 2), peers can fetch the bitmap of only some or all the other peers within their communication range. They may also select to first exchange bitmaps and then start exchanging data or interleave bitmap and data exchanges. These decisions involve a trade-off between: (i) the number of bitmaps peers retrieve, which indicates the knowledge they have about the available data (the more knowledge they have, the more efficient the downloading process will be), and (ii) how much time they have to download data.

**Analysis:** Let us assume that peers are connected for a time interval $\Delta t$ and the transmission delay is $d$. Let $\mathrm{T}_{\text{delay}}$ be the average delay for a peer to successfully transmit a bitmap (we further elaborate on $\mathrm{T}_{\text{delay}}$ in Section IV-F). If peers exchange $b$ bitmaps before they start fetching data, the average time interval $\mathrm{T}_{\text{data}}$ they will have for data fetching is:

$$T_{\text{data}} = \begin{cases} \Delta t - (T_{\text{delay}} + d) * b, & \text{if } (T_{\text{delay}} + d) * b < \Delta t \\ 0, & \text{if } (T_{\text{delay}} + d) * b \geq \Delta t \end{cases}$$

This equation shows that peers will have time for data fetching only if their encounter lasts more than the time they need for bitmap exchanges. When peers interleave their bitmap and data exchanges, after they fetch the first bitmap, they have an equal chance of sending a bitmap Interest or an Interest for data until $b$ bitmaps are exchanged. Assuming that $0 \leq b \leq \lfloor \frac{\Delta t}{T_{\text{delay}} + d} \rfloor$, the average time interval $\mathrm{T}_{\text{data}}$ that peers have for data fetching is:

$$T_{\text{data}} = \begin{cases} \Delta t - (T_{\text{delay}} + d) * b, & \text{if } T_{\text{delay}} + d < \Delta t \\ 0, & \text{if } T_{\text{delay}} + d \geq \Delta t \end{cases}$$

This equation indicates that peers will not have time for data fetching only for very small connections (i.e., they do not have enough time for a single bitmap exchange).

### *E. Data Fetching Strategy*

After exchanging advertisements, peers know what collection data is available around them and proceed to downloading. Given that peer encounters might be short-lived and the connectivity intermittent, we need to ensure that the utility of each single data transmission is maximized by prioritizing the exchange of data missing by most peers. To achieve that, we propose two variants of the RPF strategy: (i) RPF across a peer's communication range (local neighborhood), and (ii) RPF based on the history of peer encounters. Note that this DAPES component is generic and supports the deployment of *any* data fetching strategy.

**Local neighborhood RPF:** It estimates the rarity of each packet based on how many peers within the local neighborhood do not have this data. When two or more peers exchange their bitmaps, each of them computes the rarity of each packet based on how many of the received bitmaps show a packet as missing. Each peer then creates a list of missing packets; on the top of the list are packets with higher rarity value, which will be requested first. This list is specific to each set of connected peers, and expires after the peers get disconnected, thus no long term state is maintained.

**Encounter-based RPF:** It estimates the rarity of each packet based on the history of encountered peers in the swarm, providing an estimation of how many peers in the swarm do not have each packet. Peers maintain a list of the bitmap that each encountered peer has for a certain number of encounters. Whenever a peer encounters others, it updates the list to reflect the received bitmaps. The rarity of each packet is estimated based on all the bitmaps in the list. Peers request the packets with the highest rarity values first (i.e., packets that most of the bitmaps in the list show as missing).

**Trade-offs:** The two approaches offer different ways to estimate how rare a packet is. The first one allows peers to fetch data needed by as many neighbors as possible, reducing the overall number of transmissions, without requiring peers to store long term state. The second approach prioritizes data based on peers in the swarm as a whole and requires peers to store and manage local state across multiple encounters.

### *F. Data Advertisement Prioritization & Collision Mitigation*

The efficiency of the data fetching strategy (Section IV-E) depends on data advertisements (Section IV-D). Let us consider a scenario, where advertisements contain only a few missing packets, while there is a number of missing packets around peers. In this case, the data fetching process may be inefficient, since peers exchange (potentially much) less missing data than what it is actually available around them. To this end, when multiple peers encounter each other for a (potentially) short time, we need to ensure that they quickly become aware of as much available (missing) data as possible around them. To achieve that, we need mechanisms to: (i) prioritize data advertisements from peers that maximize the amount of available (missing) data that encountered peers are aware of, and (ii) mitigate transmission collisions during this process, while at the same time preserve the semantics of data advertisement prioritization.

**Data advertisement transmission prioritization:** For the transmission of the first bitmap during an encounter, the peer that has most of the data receives priority. This is useful when a peer having a few (or no) data encounters a peer that has most (or all) of the packets, so that the latter disseminates as much data as possible to the former. For each subsequent transmission, our prioritization strategy[3] is based on the number of packets each peer has that are missing from all the previously transmitted bitmaps.

**Collision mitigation:** Peers can prioritize their bitmap transmissions linearly by dividing a default transmission window by the percent of the packets they have that are missing from previously transmitted bitmaps. This, however, results in frequent collisions when peers have a similar number of packets that are missing from previous bitmaps. To mitigate that, we propose a variant of the Ethernet exponential backoff algorithm, which we call "Priority-based Exponential Backoff Algorithm (PEBA)". PEBA separates peers into groups of transmission slots[4] created through the exponential backoff algorithm, prioritizing peers that have a larger number of missing packets from all the previously transmitted bitmaps. The priority groups and the number of transmission slots are created on a per-encounter basis.

**Example:** In Figure 5, we assume that peers have a default transmission window and no collisions have occurred. When no collisions have been detected, peers prioritize their bitmap transmissions by dividing the transmission window by the percent of the packets they have that are missing from previously transmitted bitmaps. Therefore, given that A has most of the data, it schedules its transmission timer to expire before others. When peers receive A's bitmap, they cancel their current transmission and reset their timer based on how many packets they have that were missing from A's bitmap. C's timer expires before others, however, B's timer expires before hearing C's transmission. B and C collide and once they detect the collision, PEBA creates two slots.

We assume that the slots for the peers that collide are divided into two priority groups. Peers that have, at least, half of the missing packets randomly select a slot in the first group, while peers that have fewer than half of the missing packets randomly select a slot in the second group. In our example, there are six packets missing from A's bitmap. C has three, B has two, and D has one of them. Thus, C will be in the first group, while B and D will be in the second group. C transmits during the first slot, while B and D transmit during the second slot, colliding with each other. In this case, PEBA creates four slots for B and D. There are three packets missing from A's and C's bitmaps, therefore B will be in the first group, transmitting during the first or second slot, and D in the second group, transmitting during the third or fourth slot.

**Analysis:** Let us assume that there are $L$ transmission slots in total and peers are divided into $k$ priority groups, thus there are $n = \lfloor \frac{L}{k} \rfloor$ slots per group. Peers in the $jth$ group select a random slot $s$, where $j * n \leq s < (j+1) * n$. Zhu et al. [44] proved that the average backoff number before a successful transmission occurs is $N_{\text{backoff}} = \sum_{i=1}^{\infty} iP_{\text{i}}$, where $P_{\text{i}}$ is the probability that a peer has collided $i$ times before a successful transmission. The average delay for a peer to successfully transmit its bitmap is $T_{\text{delay}} = \frac{L_{\text{average}}-1}{2}\tau$, where $L_{\text{average}} = \frac{n-1}{2}$ is the peer's average contention window size and $\tau$ the slot duration.

| Peer | Bitmap |
|---|---|
| A | 1001011000 |
| B | 0110001000 |
| C | 0000000111 |
| D | 100110000 |

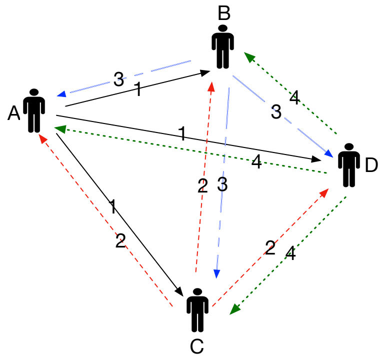


Fig. 5: Bitmap prioritization & collision mitigation example

## V. Multi-hop Communication

In this section, we present how DAPES can achieve communication over multiple hops through one or more intermediate nodes that may or may not understand the DAPES semantics.

### A. *Pure Forwarders*

Peers may meet nodes that do not understand the DAPES semantics (e.g., users that have not installed the application on their device), but understand the NDN network-layer semantics (i.e., only have an NFD instance installed). We call these nodes "pure forwarders". Pure forwarders store data transmissions they overhear in their CS, thus satisfying received requests with cached data. They also opportunistically forward Interests based on a probabilistic scheme to: (i) avoid flooding Interests across the network, and (ii) discover data available more than a single hop away. Pure forwarders wait for a random amount of time before forwarding an Interest to: (i) avoid collisions with others, and (ii) avoid unnecessary transmissions, since another node within their communication range might respond to the Interest. When they forward an Interest, but do not receive a response, pure forwarders start a suppression timer for the Interest name, not forwarding future Interests with the

[3] The prioritization scheme applies only to the transmission of bitmaps, since the RPF strategy determines the order to retrieve data. Collisions during the transmission of Interest and data packets determined by RPF are mitigated through the use of a random transmission timer by each peer.

[4] The length of the transmission slots can be based on a variety of factors. In the context of this work, we have so far considered the average size of transmitted packets and the channel state (e.g., bandwidth, loss rate).

same name until the timer expires. This timer acts as soft state information that determines whether certain data is currently reachable through a pure forwarder.

In Figure 6, the dark (A, D, F, H, K) and grey nodes (C, E, G) are interested in different file collections, while node B is a pure forwarder. We assume that A sends an Interest, which can be a discovery or a bitmap Interest, or an Interest for data. Node B receives this Interest, and further forwards it based on some probability. We assume that B forwards this Interest towards direction 1, without receiving a response, thus it starts a suppression timer for the Interest name.

### B. *Intermediate Nodes Running DAPES*

Nodes running DAPES store information about the data their neighbors have and the collections their neighbors are interested in. This helps them make adaptive forwarding decisions about the Interests they receive from others. In this way, peers reach others through one or more intermediate peers that are interested in the *same* or a *different* file collection.

**Same file collection:** Intermediate peers interested in the same file collection make forwarding decisions based on their knowledge about the available collection data across their neighbors. In Figure 6, K is a direct neighbor of A and both are downloading the same file collection. K knows whether there are peers towards direction 3 that download the same collection and what data they have. Therefore, K forwards received Interests from A only when it speculates that they can bring a response back; for example, when A requests data that K does not have, but J does, or when it is beneficial for A to learn J's bitmap (e.g., J may be able to offer multiple data packets missing from A). In our example, we assume that K speculates that forwarding A's Interest to J will not be bring a response back, therefore, K suppresses the Interest.

**Different file collections:** Intermediate peers interested in a different file collection make forwarding decisions based on messages they overhear about other collections across their neighbors. If intermediate peers have no knowledge about the requested data (e.g., have not overheard any related messages recently), they follow the probabilistic scheme of pure forwarders, suppressing Interests that do not bring data back. In Figure 6, A and F are two hops away, but can reach each other through C, who is interested in a different collection than them. When C receives A's Interest, it decides whether to forward it. For example, if C has overheard the messages between F and H, it knows that F is interested in the same collection as A, thus forwarding A's Interest towards direction 2 will likely retrieve data. Especially if C has overheard F's bitmap, it knows which packets F has, thus being able to accurately decide whether to forward A's Interest.

## VI. Experimental Evaluation

We performed a simulation study of DAPES to evaluate different design choices (Section VI-C) and compare its performance with existing IP-based solutions (Section VI-D). We also performed a DAPES feasibility study through real-world experiments in an outdoor campus setting (Section VI-E).

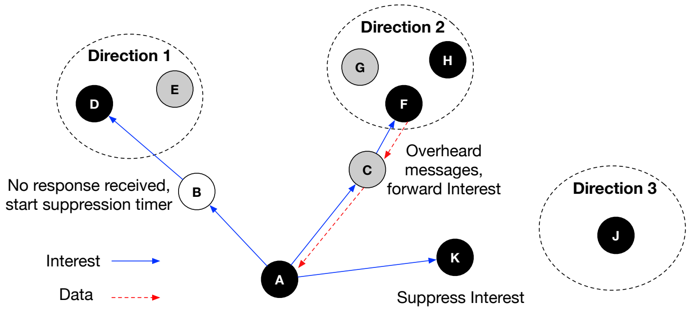


Fig. 6: DAPES multi-hop communication example

### A. *Prototype Implementation*

Our DAPES prototype consists of 5K lines of C++ code and uses the ndn-cxx library [31] to ensure compatibility with NFD. It includes three main components: i) a library that provides the fundamental data structures (e.g., metadata file) and abstractions (e.g., RPF strategy) common for peer-to-peer file sharing in wired and wireless environments, ii) a software module that adapts the library abstractions to our off-the-grid communication environment (e.g., the baseline RPF strategy to provide the different RPF flavors), and iii) an application module that uses abstractions either from the library or the adaptation module to implement the DAPES logic.

### B. *Experimental Setup*

We used a collection of image files and experimented with a variable number and size of files. We present the 90th percentile of the results collected after ten trials for simulations and real-world experiments.

*1) Simulation Experiments:* Our topology (Figure 7) consists of 4 stationary (acting as data repositories) and 40 mobile nodes. The mobile nodes randomly choose their direction and speed. The speed ranges from 2m/s to 10m/s and the direction from 0 to 2 $\pi$ (loss rate equal to 10%). Nodes communicate through IEEE 802.11b 2.4GHz (data rate of 11Mbps). We perform experiments with varying WiFi ranges, which leads to different sizes of connected peer groups over time. The 4 stationary nodes and 20 of the mobiles nodes (randomly chosen) download a *file collection of interest*. Unless otherwise noted, we used a collection of ten files (each file is 1MB and each data packet is 1KB).

**DAPES-based experiments:** We ported our DAPES prototype into the ndnSIM simulator [24]. ndnSIM features software integration with the real-world NDN software prototypes (ndn-cxx and NFD) to offer high fidelity of simulation results. In our topology (Figure 7), we randomly select 10 nodes to act as pure forwarders and the remaining 10 nodes understand the DAPES semantics and act as intermediate nodes. Peers use a transmission window of 20ms and select a random value within this window for every transmission other than bitmap transmissions, which are prioritized. Unless otherwise noted, peers use the local neighborhood RPF strategy, interleave data fetching with data advertisements, and fetch advertisements from all the peers within their range. They also communicate over multiple hops (unless otherwise noted the probability of intermediate nodes to forward an Interest is 20% to ensure message reachability, but also avoid extensive flooding).

**IP-based experiments:** We compare DAPES to two IP-based peer-to-peer file sharing solutions for MANET; Bithoc [17], [37] and Ekta [35]. Bithoc peers perform periodic scoped flooding of "HELLO" messages to discover others and the data they have. They separate others into "close" (at most two hops away) and "far" (more than two hops away) neighbors. Peers follow an RPF strategy to fetch data from close neighbors, while they fetch data not available in their nearby neighborhood from far neighbors. Bithoc uses DSDV [34] as the underlying routing protocol and TCP over IP for reliable delivery. Ekta offers a Distributed Hash Table (DHT) substrate for the search of data objects in MANET by integrating the DHT protocol operations with DSR [15] at the network layer. Ekta uses UDP over IP as the transport layer protocol. Following the setup of the DAPES experiments for a fair comparison, in our topology (Figure 7), we randomly select 10 nodes to act as forwarders and the last 10 nodes understand the Bithoc and Ekta semantics. All of these 20 nodes forward received packets based on their routing tables.

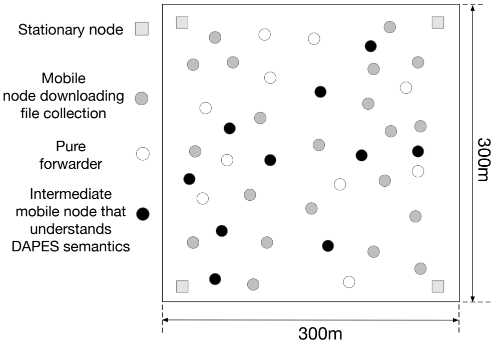

Stationary node
Mobile node downloading file collection
Pure forwarder
Intermediate mobile node that understands DAPES semantics
300m
300m

Fig. 7: Simulation topology snapshot

**Evaluation metrics:** We present results (Sections VI-C and VI-D) for the following metrics: (i) *file collection download time:* the average time needed for each of the 20 mobile and 4 stationary nodes to download the file collection of interest, and (ii) *transmissions (overhead):* the number of packets transmitted by all the 44 nodes (24 nodes that download the file collection of interest and the 20 intermediate nodes) for the downloading of the collection of interest. For DAPES, the overhead includes the discovery Interests and data, bitmap Interests and data, and the Interest/data packets transmitted for the file collection sharing (including the forwarding transmissions by intermediate nodes). For Bithoc, the overhead includes the packets generated by DSDV, the application-layer flooding, and the TCP overhead for the collection retrieval. For Ekta, the overhead includes the packets generated by DSR for route discovery and maintenance, messages among peers on the DHT to find data packets, and the packets needed to retrieve the file collection.

*2) Real-world Experiments:* We used 5 MacBooks (macOS 10.13), each equipped with an 1.7GHz Intel i7 processor and 8 GB of memory. We ran NDN on top of IEEE 802.11b 2.4GHz (each MacBook had a WiFi range of about 50m). Peers interleaved data fetching with data advertisements and fetched advertisements from all the entities within their range. Peers also used the RPF strategy across their local neighborhood.

We experimented with three different scenarios in an outdoor campus setting. In the first one (Figure 8a), peer A generates a file collection. D acts as a data carrier that fetches the collection from A and carries it to other network segments, where peers B and C fetch it. In the second one (Figure 8b), C generates a collection. The repo downloads the collection from C, while A and B download the collection from the repo. In the third one (Figure 8c), A generates a collection that shares with B, C, and D (peers are moving across an area with no infrastructure). To demonstrate how DAPES maximizes the utility of the transmitted data and multi-hop communication, in this scenario, there are moments that all the peers are disconnected and moments that they are within the communication range of each other.

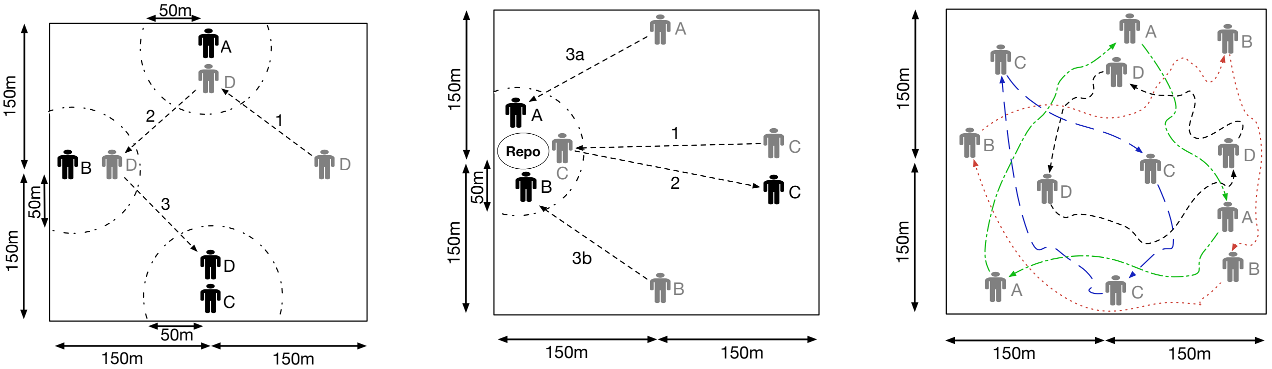
### C. DAPES Design Trade-offs

**Data fetching strategy:** In Figure 9a, we present the download time for the encounter-based and local neighborhood RPF strategies when peers first fetch the bitmap of all the others within their communication range and then share data. The results show that the local neighborhood strategy performs about 12-14% better than the encounter-based. The former strategy focuses on retrieving the data missing by most of the peers within their current neighborhood, while the latter also considers previous encounters among peers that might not be within the communication range of each other anymore. As a result, fewer transmissions take place among peers when the local neighborhood strategy is used (Figure 9b).

The results also show that when peers start their downloading process with a random rather than the same packet of the file collection, they are able to download the collection about 11-15% faster. Starting with a random packet in the file collection helps peers retrieve different blocks of data, thus increasing the diversity of the disseminated data. Note that as we increase the WiFi range (more peers are directly connected to each other), the download time decreases at a slower rate. We conclude that this is due to collisions, given that the number of transmissions for both strategies increases with the WiFi range as shown in Figure 9b.

**Collision mitigation:** Figure 9b shows the number of transmissions for both flavors of the RPF strategy with and without PEBA (Section IV-F). Without PEBA, both strategies result in a large number of transmissions as the WiFi range increases. This is due to collisions for the bitmap transmissions; to prioritize bitmap transmissions, peers divide their transmission window by the percent of packets they have, which are missing from previously transmitted bitmaps. As the WiFi range increases, more peers are directly connected. This results in more peers that have similar data and, as a consequence, schedule their transmissions very close to each other. On the other hand, PEBA reduces the number of transmissions by 22-28%, since it mitigates bitmap transmission collisions through an exponential backoff mechanism, which at the same time preserves the bitmap prioritization semantics.

**Data advertisement exchange strategy:** In Figure 9c, we present the download time when peers exchange their bitmaps first and then download data for a varying number of ex-

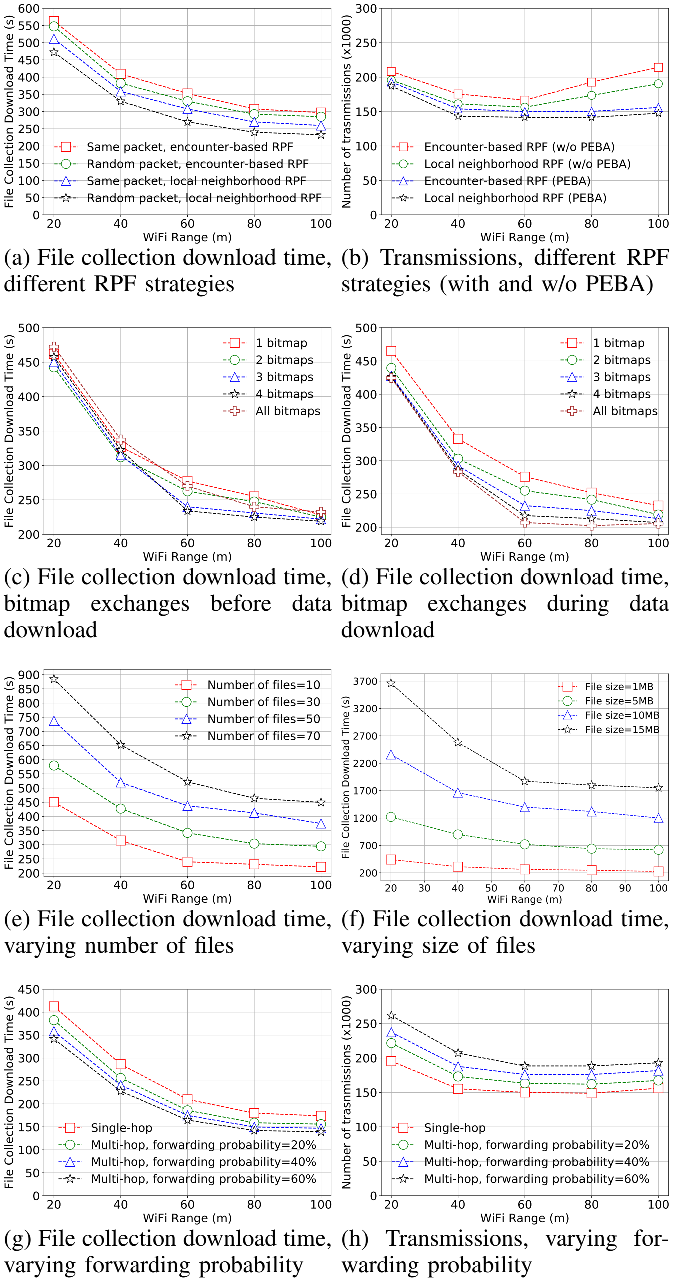


(a) Data sharing through a carrier

(b) Data sharing through a repository

(c) Data sharing among moving nodes

Fig. 8: Real-world experimental scenarios



(a) File collection download time, different RPF strategies

(b) Transmissions, different RPF strategies (with and w/o PEBA)

(c) File collection download time, bitmap exchanges before data download

(d) File collection download time, bitmap exchanges during data download

(e) File collection download time, varying number of files

(f) File collection download time, varying size of files

(g) File collection download time, varying forwarding probability

(h) Transmissions, varying forwarding probability

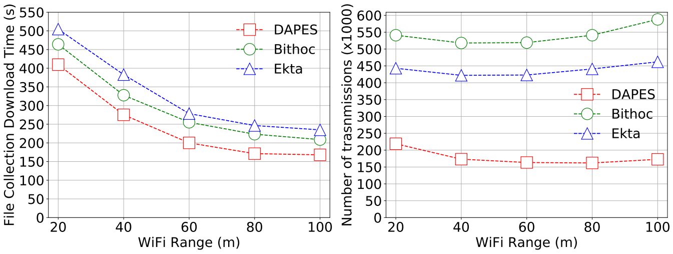
Fig. 9: DAPES design trade-off results

changed bitmaps. The results show that peers minimize the download time when they have enough knowledge about the available data around them, so that the RPF strategy can make effective decisions (2-3 bitmaps for shorter and 4 bitmaps for longer WiFi ranges respectively). Peers move away from each other if they spend too much time exchanging bitmaps before downloading data (e.g., in the illustrated “all bitmaps” case, where peers fetch the bitmap of every other peer within their communication range). This conclusion verifies our analysis in Section IV-D.

In Figure 9d, we present the download time when peers interleave their bitmap and data exchanges. The results demonstrate the benefit from fetching data as soon as peers have any knowledge about the available data around them. As they collect more bitmaps, the RPF strategy becomes more effective and peers download data faster. This interleaved bitmap and data fetching strategy results in 16-23% shorter download times than the strategy of fetching bitmaps first and then data.

**Variable number and size of files:** In Figures 9e and 9f, we present the download time for a varying number of files per collection (each file is 1MB) and varying sizes of collection files (each collection has ten files) respectively. As expected, the download time increases with the total amount of data to be shared. The results demonstrate that the properties of DAPES hold as the collection size grows.

**Impact of intermediate nodes:** In Figure 9g, we present the download time for a varying forwarding probability by intermediate nodes, while Figure 9h shows the number of packets transmitted for the file collection retrieval. When pure forwarders and intermediate DAPES nodes with no knowledge about the requested data forward 20-60% of received Interests, the collection download time decreases by 12-23% compared to the results for the DAPES single-hop design. On the other hand, the overhead (packet transmissions) increases by 14-38%. Overall, the results show that it is adequate for pure forwarders and intermediate DAPES nodes with no knowledge about the requested data to forward 20-40% of the received Interest. Forwarding a larger amount of Interests results in little performance gain and substantial overhead increase.



(a) Download time (b) Transmissions

Fig. 10: Comparison to IP-based solutions

### *D. Comparison to IP-based Solutions*

**File collection download time:** In Figure 10a, we present the download time results, which show that DAPES achieves 15-27% and 19-33% lower download times than Bithoc and Ekta respectively. Bithoc and Ekta identify individual receivers (based on their IP address) for each packet they send. Even if multiple peers missing common data are within the communication range of the sender, a separate packet has to be sent to each one of them. Paths to each peer need to be established first, then discover what data a peer has, and then begin data retrieval. On the other hand, the semantically meaningful naming of DAPES enables peers to identify the data missing by most of the peers around them, maximizing the utility of transmissions[5]. DAPES decouples data sharing from the location of nodes within the network; data requests can be satisfied with data close to the requester, while intermediate nodes can satisfy Interests with cached data.

**Transmissions:** Figure 10b presents the number of transmissions for each solution. DAPES results in 62-71% and 50-59% lower overheads than Bithoc and Ekta respectively. Bithoc relies on proactive routing to maintain routes towards peers, and application-layer messages to discover what data each peer has. Due to intermittent connectivity, established routes break and the TCP performance degrades over multiple wireless hops [14]. Ekta is based on reactive routing, resulting in lower overheads than Bithoc (routes are maintained on-demand). In DAPES, each data packet is useful to multiple peers, while intermediate nodes forward or suppress received Interests based on the data available around them. This results in accurate forwarding decisions (83% of the forwarded Interests successfully brought data back) and low overheads.

### *E. Real-World Feasibility Study*

In Table I, we present the results for each scenario of Figure 8. Overall, these results verify the conclusions of our simulation study. DAPES can offer with a single transmission data needed by multiple peers, while multi-hop communication comes with its own cost in terms of system load, since peers need to store information about the data available around them.

In the first scenario (Figure 8a), the communication involves only two parties (the data carrier and a peer in each connected group), therefore, more time and transmissions are needed for all the entities to download the file collection. In the second scenario (Figure 8b), peers A and B fetch the file collection from the repository at the same time. Data requested by either A or B can satisfy both, therefore, the collection can be downloaded faster with fewer transmissions. In the third scenario (Figure 8c), peers take advantage of the time that are within the range of each other, and the multi-hop communication to further optimize data sharing. In this scenario, the results also indicate that the system load, in terms of memory consumption, page faults, system calls, and context switches per second, increases. This is due to the greater amount of multi-hop communication among peers, which requires peers to maintain information about the available data around them.

| Scenario | Download Time (s) | Number of Transmissions | Memory Overhead (MB) |
|---|---|---|---|
| 1 | 454 | 30,841 | 14.75 |
| 2 | 418 | 24,243 | 14.65 |
| 3 | 213 | 16,102 | 18.65 |

| Scenario | Context Switches | System Calls | Page Faults |
|---|---|---|---|
| 1 | 56,413 | 214,313 | 4,742 |
| 2 | 53,472 | 202,542 | 4,683 |
| 3 | 46,619 | 186,548 | 4,274 |

TABLE I: Real-world feasibility study results

## VII. Conclusion & Future Work

In this paper, we presented DAPES, a data-centric design for off-the-grid peer-to-peer file sharing. DAPES offers mechanisms to maximize the utility of transmissions under intermittent connectivity and short-lived connections. It also achieves communication over multiple wireless hops by building short-lived knowledge about the data available around peers.

While DAPES is off to a promising start, we plan to investigate a number of open issues in the future. First, we plan to conduct further real-world and simulation experiments, where peers share large numbers of file collections simultaneously. This will help us "stress-test" the scalability limits of our multi-hop communication design (e.g., amount of information that peers need to maintain) and our collision mitigation mechanism. Second, we plan to investigate the impact of having intermediate DAPES peers carry on behalf of others received Interests and data packets in their PIT and CS respectively. This direction will explore an off-the-grid file sharing approach that resembles more to Delay-Tolerant Networking (DTN) [5], [11] rather than MANET. Third, previous work [19], [39], [40] has demonstrated that the IEEE 802.11 MAC protocol may suffer from low throughput and high error rates in MANET communication scenarios. To this end, we plan to investigate the feasibility of data-centricity starting from the MAC layer of the network architecture all the way up to the application layer, and use DAPES as a driver example to investigate the impact of a data-centric MAC layer protocol [10] on applications.

[5] Note that UDP multicast satisfies multiple communicating parties through a single packet transmission, however, it may still require the configuration of a multicast IP address and a multicast routing protocol for communication over multiple hops. It also does not receive the benefits of stateful forwarding, data-centric security and caching from the underlying network.

## References

[1] NDN Packet Format Specification. https://named-data.net/doc/ndn-tlv/.

[2] Alexander Afanasyev, Junxiao Shi, et al. NFD Developer's Guide. Tech. Rep. NDN-0021, NDN, 2015.

[3] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications*, 50:148–158, 2015.

[4] Marica Amadeo, Antonella Molinaro, and Giuseppe Ruggeri. E-chanet: Routing, forwarding and transport in information-centric multihop wireless networks. *Computer communications*, 36(7):792–803, 2013.

[5] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, 2003.

[6] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[7] Brian P Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T Sakai. Ieee 802.11 wireless local area networks. *IEEE Communications magazine*, 35(9):116–126, 1997.

[8] Andrea Detti, Bruno Ricci, and Nicola Blefari-Melazzi. Peer-to-peer live adaptive video streaming for information centric cellular networks. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 3583–3588. IEEE, 2013.

[9] Andrea Detti, Bruno Ricci, and Nicola Blefari-Melazzi. Mobile peer-to-peer video streaming over information-centric networks. *Computer Networks*, 81:272–288, 2015.

[10] Mohammed Elbadry et al. Poster: A raspberry pi based data-centric mac for robust multicast in vehicular network. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 714–716. ACM, 2018.

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

[12] Neelakantam Gaddam and Anupama Potluri. Study of bittorrent for file sharing in ad hoc networks. In *2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–6. IEEE, 2009.

[13] Guoyou He. Destination-sequenced distance vector protocol. *Networking Laboratory, Helsinki University of Technology*, 2002.

[14] Gavin Holland and Nitin Vaidya. Analysis of tcp performance over mobile ad hoc networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, pages 219–230, New York, NY, USA, 1999. ACM.

[15] David B Johnson, David A Maltz, Josh Broch, et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, 5:139–172, 2001.

[16] Alexander Klemm, Christoph Lindemann, and Oliver P Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, volume 4, pages 2758–2763. IEEE, 2003.

[17] Amir Krifa, Mohamed Karim Sbai, Chadi Barakat, and Thierry Turletti. Bithoc: A content sharing application for wireless ad hoc networks. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–3. IEEE, 2009.

[18] Uichin Lee et al. Code torrent: content distribution using network coding in vanet. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5. ACM, 2006.

[19] Jinyang Li, Charles Blake, Douglas SJ De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 61–69. ACM, 2001.

[20] Tianxiang Li, Zhaoning Kong, Spyridon Mastorakis, and Lixia Zhang. Distributed dataset synchronization in disruptive networks. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2019.

[21] Jared Lindblom, M Huang, Jeff Burke, and Lixia Zhang. Filesync/ndn: Peer-to-peer file sync over named data networking. *NDN Technical Report NDN-0012*, 2013.

[22] Fabio Malabocchia, Romeo Corgiolu, Maurizio Martina, Andrea Detti, Bruno Ricci, and Nicola Blefari-Melazzi. Using information centric networking for mobile devices cooperation at the network edge. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE, 2015.

[23] Spyridon Mastorakis, Alexander Afanasyev, Yingdi Yu, and Lixia Zhang. ntorrent: Peer-to-peer file sharing in named data networking. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10. IEEE, 2017.

[24] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. On the evolution of ndnsim: An open-source simulator for ndn experimentation. *ACM SIGCOMM Computer Communication Review*, 47(3):19–33, 2017.

[25] AJ McAuley and K Manousakis. Self-configuring networks. In *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No. 00CH37155)*, volume 1, pages 315–319. IEEE, 2000.

[26] Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8. ACM, 2010.

[27] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[28] Archan Misra, Subir Das, Anthony McAuley, and Sajal K Das. Autoconfiguration, registration, and mobility management for pervasive computing. *IEEE Personal Communications*, 8(4):24–31, 2001.

[29] Luca Muscariello, Giovanna Carofiglio, and Massimo Gallo. Bandwidth and storage sharing performance in information centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 26–31. ACM, 2011.

[30] Alok Nandan, Shirshanka Das, Giovanni Pau, Mario Gerla, and MY Sanadidi. Co-operative downloading in vehicular ad-hoc wireless networks. In *Second Annual Conference on Wireless On-demand Network Systems and Services*, pages 32–41. IEEE, 2005.

[31] NDN Team. ndn-cxx. http://named-data.net/doc/ndn-cxx.

[32] Sanket Nesargi and Ravi Prakash. Manetconf: Configuration of hosts in a mobile ad hoc network. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1059–1068. IEEE, 2002.

[33] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[34] Charles E Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *ACM SIGCOMM computer communication review*, volume 24, pages 234–244. ACM, 1994.

[35] Himabindu Pucha, Saumitra M Das, and Y Charlie Hu. Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In *Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 163–173. IEEE, 2004.

[36] Sundaram Rajagopalan and Chien-Chung Shen. A cross-layer decentralized bittorrent for mobile ad hoc networks. In *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems-Workshops*, pages 1–10. IEEE, 2006.

[37] Mohamed Karim Sbai, Chadi Barakat, Jaeyoung Choi, Anwar Al Hamra, and Thierry Turletti. Bithoc: Bittorrent for wireless ad hoc networks. *Project-Team Planete, INRIA Sophia Antipolis, France*, 2008.

[38] Matteo Varvello, Ivica Rimac, Uichin Lee, Lloyd Greenwald, and Volker Hilt. On the design of content-centric manets. In *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, pages 1–8. IEEE, 2011.

[39] Shugong Xu and Tarek Saadawi. Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks? *IEEE communications Magazine*, 39(6):130–137, 2001.

[40] Shugong Xu and Tarek Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. *Computer Networks*, 38(4):531–548, 2002.

[41] Hao Yang, Haiyun Luo, Fan Ye, SW Lu, and Lixia Zhang. Security in mobile ad hoc networks: challenges and solutions. 2004.

[42] Lixia Zhang et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[43] Zhiyi Zhang et al. An overview of security support in named data networking. *IEEE Communications Magazine*, 56(11):62–68, 2018.

[44] Yi-Hua Zhu, Xian-Zhong Tian, and Jun Zheng. Performance analysis of the binary exponential backoff algorithm for ieee 802.11 based mobile ad hoc networks. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2011.