SoftwareX 9 (2019) 55–60

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

VECTORS — VidEo Communication Through Opportunistic Relays and Scalable video coding

Abhishek Thakur [*], Arnav Dhamija, Tejeshwar Reddy G.

BITS Pilani, Hyderabad Campus, Jawahar Nagar, Hyderabad, Telangana, 500078, India

A R T I C L E I N F O

Article history:
Received 31 May 2018
Received in revised form 26 December 2018
Accepted 27 December 2018

Keywords:
User-generated video content
Opportunistic communication
User mobility
Disruption tolerant network

A B S T R A C T

Crowd-sourced video distribution is frequently of interest in the local vicinity. In this paper, we present a software platform called VECTORS to transfer videos over opportunistic networks with adaptive quality encoding to achieve reasonable delay bounds. The video segments are transmitted between source and destination in a delay tolerant manner using the Nearby Connections Android library. VECTORS can be applied to multiple domains including farm monitoring, wildlife, environmental tracking, or disaster response scenarios. In addition to the software architecture of VECTORS, we also discuss basic results for the trial runs conducted within our institute and provide an empirical analysis of proposed architecture using simulation.

Code metadata

Current code version	v9
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_67
Legal Code License	GPLv2
Code versioning system used	git
Software code languages, tools, and services used	Java, Android, Linux Shell, C, C++
Compilation requirements, operating environments & dependencies	Android 8.0 SDK, Google Play Services, Android Nearby Connections Library, Ubuntu 16.04
Link to developer manual	https://github.com/swifiic/Vectors/wiki
Support email for questions	apps4rural@gmail.com

Software metadata

Current software version	v1.4.0
Permanent link to executables of this version	https://play.google.com/store/apps/details?id=in.swifiic.vectors
Legal Software License	GPLv2
Computing platforms/Operating Systems	Android, Linux
Installation requirements & dependencies	Android 7.0 and above, SHM 12.4, JSVM, Ubuntu 16.04
Link to user manual	https://github.com/swifiic/Vectors/wiki
Support email for questions	apps4rural@gmail.com

1. Motivation and significance

In recent years, user-generated video content has increased several-fold [1]. The content is typically captured from mobile devices and uploaded to servers for subsequent viewing. These solutions do not work in the absence of network infrastructure. Delay and Disruption Tolerant Networks (DTNs) [2] can be applied in

[*] Corresponding author.
E-mail addresses: abhishek@hyderabad.bits-pilani.ac.in (A. Thakur), arnav.dhamija@gmail.com (A. Dhamija), tejeshwarreddy1996@gmail.com (Tejeshwar Reddy G.).

https://doi.org/10.1016/j.softx.2018.12.006

such scenarios for communication. Opportunistic Networks (OppNets) are a special case of DTNs in which the contact patterns are unpredictable. However, existing approaches that send video at fixed quality may suffer from high delay or drop rates. If the video flow is not adaptive, then the network may not be optimally utilized.

Common approaches for streaming video over the Internet (e.g., DASH/TCP) rely on the end-to-end transmission for adapting the quality of the video to the network conditions. These approaches perform well when the round-trip times are below a second. However, such adaptations will not suit OppNets, primarily on two counts: (1) the feedback from the destination can take an undeterministically long time to reach the source, and (2) the feedback about the acknowledged packets from the destination may be lost.

OppNets create replicas of the data to reduce delivery delay and improve delivery probability. However, the creation of too many replicas can overload the network. Hence, we need to devise a solution that delivers perceivably good quality video without overloading the network. The OppNet should be able to improve or reduce the quality of the delivered video depending the on the network resources (e.g. buffer space, number of nodes, frequency of contacts).

In this paper, we present "**V**id**E**o **C**ommunication **T**hrough **O**pportunistic **R**elays and **S**calable video coding" (VECTORS). VECTORS is a software platform for carrying scalable video over OppNets, deployed using Android devices. Scalable Video Coding (SVC) [3,4] compresses the segments of video into payloads at different SVC layers. Section 1.1 discusses the SVC compression algorithm and provides an overview of DTN routing. The video source in VECTORS adapts the number of SVC layers being transmitted, to optimize the delivered video quality as discussed in Section 2. Section 3 covers experimental results and empirical analysis using simulation. Subsequent parts cover the possible impact of VECTORS on other domains, its limitations and future work.

1.1. Background and prior work

SVC compresses the video into multiple layers. The base layer may have a lower frame rate, lower resolution, higher quantization error, and lower bits-per-sample. One or more enhancement layers can improve video quality. The improvements can be temporal (higher frame rate), spatial (better resolution), quantization (lower loss), or bit-depth. Each SVC layer may be transmitted separately. The destination needs the base layer and contiguous enhancement layers for decoding. For example, if the base layer is not delivered, then the particular segment cannot be decoded even if the destination receives all the other enhancement layers (see Table 1 for definition of the terms).

OppNet routing uses the principle of Store-Carry-Forward to deliver the payload from source to destination. OppNets typically use multi-copy routing to improve the probability of delivery and to reduce the delivery delay. Epidemic routing [5] allows an uncontrolled replication of payloads and provides the best results when the network is lightly loaded. As network load increases, the creation of additional copies can lead to congestion, which may negatively impact the proportion of payloads delivered at the destination. Multiple routing protocols have been proposed to control the overheads of creating several copies. Spray-And-Wait [6] (SNW) uses a simple approach wherein the source node controls the maximum number of copies that will exist on the network. In Binary SNW protocol, once a payload is relayed (say from A to B), both the nodes will adjust the copy-counts by half. If A (the origin of relay) had L as copy count, it retains $\lceil L/2 \rceil$ copies while B gets $\lfloor L/2 \rfloor$ copies. If the copy count of a payload falls to one, then the node cannot send the payload to any other node, except for directly sending it to the destination.

In most OppNet deployments, it is frequently the case that some message payloads may not reach the destination. Transferring standard H.264/H.265 encoded video may result in several segments of video being lost entirely. On the other hand, SVC encoded video is suitable for transferring over OppNets, as SVC does not require all the layers of the video to be transmitted for decoding. As more nodes come into contact with the destination node, there is a greater chance for more SVC enhancement layers to be transferred and the decoded video quality to improve.

Lenas [7], Morgenroth [8] and Blanchet [9] have demonstrated video streaming over DTNs for controlled network settings. These demonstrations rely on specialized nodes (e.g., satellites and static nodes with round-trip delay below 2 s) to help complete the video flow over DTN. However, these approaches do not work for OppNets where the contact patterns are unpredictable.

As done in [8], we experimented with IBR-DTN [10] for delay tolerant communication between Android devices. In the absence of access points, IBR-DTN can work by creating an ad-hoc Wi-Fi Direct [11] connection. However, Wi-Fi Direct on Android requires human interaction to approve each new connection. This makes it inconvenient for real-world deployments in which users expect the devices to connect automatically.

2. Software description

The software for VECTORS has three main components: (1) the adaptive video compressor and extractor at the source, (2) the VECTORS Android application for the OppNet, and (3) the video combiner and decoder at the destination.

2.1. Software architecture

2.1.1. Source

The source node uses a cronjob to run a script to capture the raw video. It subsequently encodes the captured video using SHM. The SVC layers are extracted using the "ExtractAddLS" executable. Based on the acknowledgments that have been received, the "push_at_src.sh" script adjusts the number of SVC layers to be sent. It checks for acknowledgments for segments transmitted in the previous 6, 12 and 24 h. If all the three segments are acknowledged, then it increases the number of SVC layers that can be sent. If none of them are acknowledged, then it reduces the number of SVC layers. If some of the segments are acknowledged, then it does an additive increase or multiplicative decrease for the number of layers being transmitted. For each payload transmitted, the source node assigns the maximum number of replicas (L) that may exist on VECTORS nodes.

2.1.2. Destination

The destination node downloads all the payloads from the attached mobile device using a cronjob. It also uploads an acknowledgment file including the timestamp for the newly received payloads.

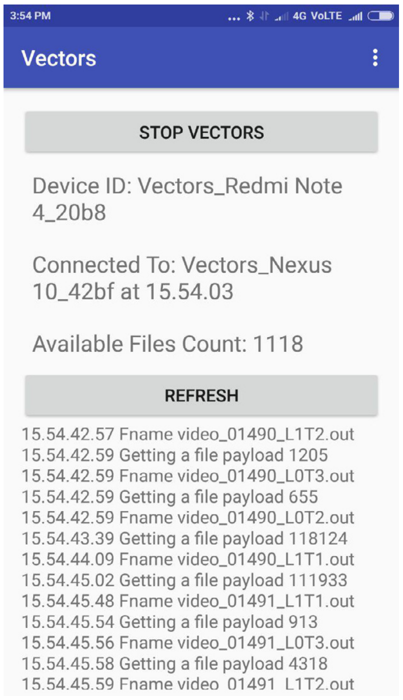
2.1.3. VECTORS Android App

The Android application implementing VECTORS OppNet has three main components:

- Startup Screen — this is a simple user interface to enable/disable VECTORS on the device. Apart from this, it also displays the communication state and logs as shown in Fig. 1.
- VECTORS Service — this is a background service implementing the relay of payloads between nodes using the state diagram illustrated in Fig. 2.
- StorageModule — this implements the storage and management of payload within the node. This module also deletes the payloads which have exceeded their Time-To-Live values.

Table 1
Key terminology for video communication over OppNet.

Term	Definition
Acknowledgment	This is a list of payload-identifiers that have been received by the destination. Nodes delete local copies for such payloads.
Copy-count (L)	The number of copies (replicas) that can be created for the payload. The source node sets the initial value of L.
Destination	The node that will decode the video segments.
Node	A device participating in the VECTORS network. This may be mobile or static.
Opportunistic Contact	When two nodes are in the vicinity of each other and capable of exchanging data.
Payload	The unit of application-level data being communicated through OppNets.
Relay	Transfer of a payload to a node that does not have it. This can create additional copies in OppNets.
Segment	The interval of video on which compression is done at the source.
Source	The node that is capturing and compressing the video segments.
SVC-Layers	A video segment will generate multiple payloads for transmission. It includes base-layer and multiple enhancement layers.
TTL	Time-to-live — the duration for which the OppNet will store the payload (from creation time).



Device ID: Vectors_Redmi Note 4_20b8

Connected To: Vectors_Nexus 10_42bf at 15.54.03

Available Files Count: 1118

```
15.54.42.57 Fname video_01490_L1T2.out
15.54.42.59 Getting a file payload 1205
15.54.42.59 Fname video_01490_L0T3.out
15.54.42.59 Getting a file payload 655
15.54.42.59 Fname video_01490_L0T2.out
15.54.43.39 Getting a file payload 118124
15.54.44.09 Fname video_01490_L1T1.out
15.54.45.02 Getting a file payload 111933
15.54.45.48 Fname video_01491_L1T1.out
15.54.45.54 Getting a file payload 913
15.54.45.56 Fname video_01491_L0T3.out
15.54.45.58 Getting a file payload 4318
15.54.45.59 Fname video_01491_L1T2.out
```

Fig. 1. User Interface for the VECTORS application on Android nodes.

Apart from the above components, the VECTORS Android app has additional classes to serialize/deserialize the metadata and acknowledgement files. It also handles life-cycle events such as restarting the VECTORS Service after the phone boots up.

2.2. VECTORS connection states

OppNet nodes, in this case, Android smartphones, are connected to each other using the VECTORS Android app. As the connections are opportunistic, the connection could be terminated any time due to the two nodes moving out of range (denoted by dotted lines in Fig. 2). Hence, the payload transfers can be considered to be best-effort. Our goal is to maximize the number of successful SVC layer transferred between two connected nodes over the opportunistic contact.

Nodes send video payloads only if the value of L is greater than one (“Transfer files” state in Fig. 2). Acknowledgments are shared to all the nodes and are not constrained by copy counts.

Discover and Connect

Nodes broadcast their auto-generated unique ID using Nearby Connections [12]. On discovery of another node, it checks the ID of the remote node to decide whether a connection should be established. If the connection is successfully established, then the node moves to Connection Initialized state. Otherwise, the node remains in the Discovery state and then it attempts to connect to other visible nodes.

Occasionally, two nodes can have the same set of payloads because of prior contacts. To avoid connections in such cases where no payloads will be transferred, the connection is rejected for nodes with successful contacts in the last five minutes. Node mobility and other environmental conditions may cause the connection to fail, despite a node initiating the connection and the other node successfully accepting it.

Endpoints Connected

Once both the nodes connect to each other, they exchange a set of control messages before transferring payloads. Each control message contains a four-byte header indicating the type of message followed by the message contents.

First, they exchange the Acknowledgement (ACK) from the video destination. If the received ACK is newer than the current one on the node, then it replaces the older ACK. Payloads acknowledged in the incoming ACK are deleted on the intermediate nodes.

In the next step, the nodes exchange a list of the payloads available with each other. From the list received, the node identifies the set of payloads to be requested. Based on the requested list, each payload is successively transferred with its VECTORS metadata. It transmits payloads in the descending order of their copy-count. The VECTORS metadata tracks the present copy-count of the payload and the nodes the payload has traversed. After successfully transferring the payloads, the metadata is updated and the copy-count is halved at both the nodes.

Connection Termination

Once a node has successfully received all the files, it sends a Connection Termination control message to the other node. Both nodes “gracefully” disconnect on receiving this control message from each other.

As the network is ad-hoc and the nodes are mobile, there is a high chance of unplanned disconnections. For such unplanned disconnections, not all the payloads are transferred successfully. In this case, the nodes revert to the Discovery state without caching each other’s ID in the recently connected list. This allows the pair of nodes additional attempts to exchange these payloads on reconnection.

2.3. Test setup

A deployment of VECTORS consists of a source and a destination node as shown in Fig. 3. Multiple Android devices running the VECTORS application are used to create the opportunistic network.

The software at the source node records the raw video segments for five minutes before compressing it using SHM [13] to create multiple SVC layers. The source node extracts each SVC layer as a separate payload. The extraction information is also transmitted to the destination using VECTORS.

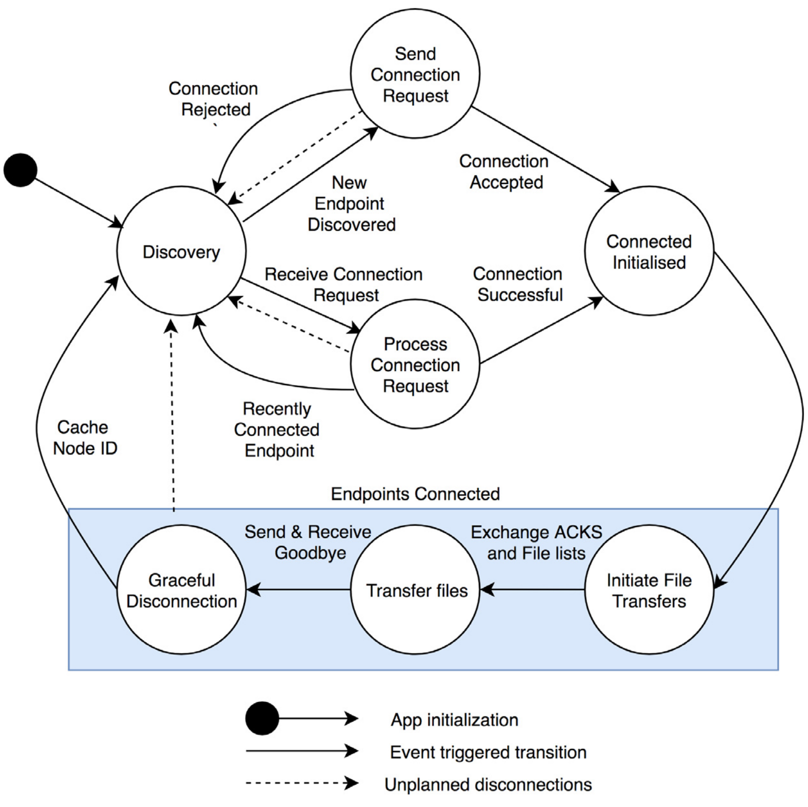


Fig. 2. VECTORS connection states (VECTORS Service).

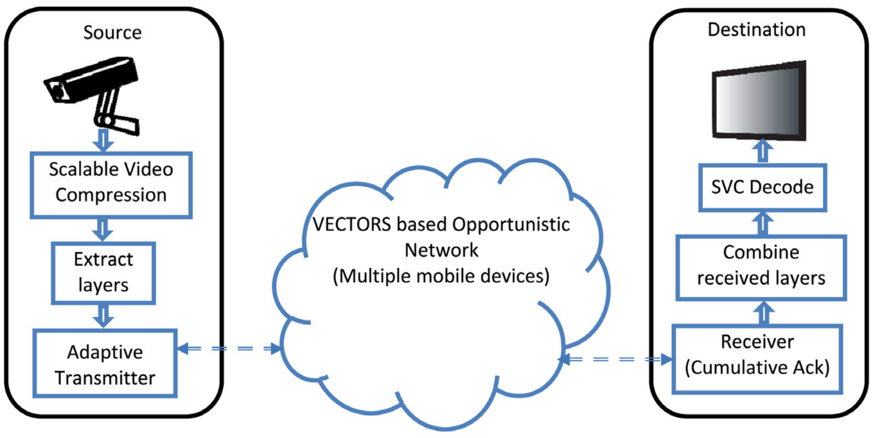

Source
Scalable Video Compression
Extract layers
Adaptive Transmitter
VECTORS based Opportunistic Network (Multiple mobile devices)
Destination
SVC Decode
Combine received layers
Receiver (Cumulative Ack)

Fig. 3. Data flow for SVC content using VECTORS.

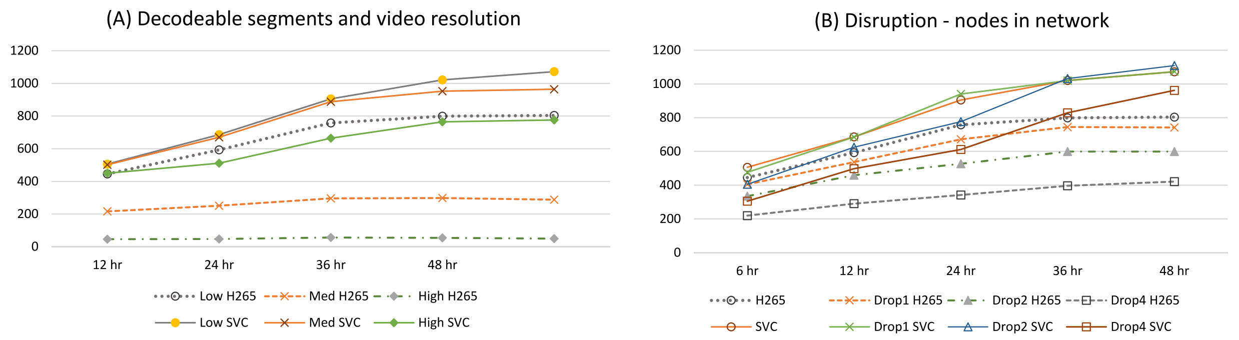


Fig. 4. Count of contacts between nodes.

The software at the destination (the decoder) combines the received payloads using the extraction information. Note that the lower layers and the extraction information are necessary to decode higher SVC layers. The destination software also generates acknowledgment for all the payloads that it receives.

For our deployment, both the source and destination are Linux computers as SHM does not have an Android port. We connect an Android device to both the source and the destination. These computers use the Android Debug Bridge (ADB) to transfer the payloads to/from mobile phones over the USB interface.

3. Illustrative examples and analysis

For the initial experiments with VECTORS, both the source and destination were installed in a lab. Following this, we moved the source and destination to different corners of our academic building. In this stage, we used our personal devices for testing the deployment. This is akin to firefighters responding to an emergency situation in the corner of a building. After resolving the issues related to the integration of encoding/decoding blocks and adaptation, we moved the setup to two different locations within our campus. The nodes are separated by a walking distance of 1.2 km. We invited volunteers to install the VECTORS app on their devices. Over a two-week interval, 15 active nodes (each with more than 20 opportunistic contacts), helped complete the network.

3.1. Empirical analysis

Since VECTORS operates in an open environment, it is not feasible to recreate identical experimental setup for different approaches to transmit video. To compare the performance of adaptive SVC transmission, we simulated different network load and node behaviors in the ONE simulator [14]. The simulation uses the contacts logged for the most active week of the deployment in Fig. 4. We compressed the captured video in raw format (YUV), at different video resolutions (Low - 320 $\times$ 240, Medium - 640 $\times$ 480 and High - 1280 $\times$ 960). For payload size, the simulation uses the file sizes for H265 (standard compression, without scalability) video and SVC layers (using SHM). Fig. 4 shows the number of successfully delivered segments on Y axis, for various TTL values on X axis. For Fig. 4(A), we ran simulations for different video resolutions, with and without adaptive-SVC transmission. In Fig. 4(B) we simulated higher disruption by removing nodes with maximum contacts, for Low-resolution video transmission. One, two and four nodes with highest contacts were dropped from the simulation. Adaptive SVC performs better than non-SVC as the nodes are removed, showing that it is more resilient to disruptions in the network. For brevity, further analysis of figures is left out of this paper.

4. Impact

Depending on the mobility patterns of the devices and the distance between source and destination, the delays can vary from an order of minutes to order of hours. In a controlled setup, it is possible to transmit video with delays of the order of a few seconds. This can provide an affordable alternative to expensive network links (e.g. leasing satellite bandwidth) when real-time communication is not required. We discuss three sample scenarios below. The low cost of Android devices and ease of deployment makes VECTORS useful for a wide variety of other applications as well.

Wildlife monitoring: In the past, researchers monitored wildlife using cameras from which they would manually extract the data. This is a time-consuming and labor-intensive approach. Prior research (Zebranet [15]) has utilized location and other contact details to be collected by using motes embedded in collars of animals. VECTORS can help simplify this labor-intensive operation by automatically streaming the video of the areas where these animals roam. A solar powered, static video source can be deployed in the wild for this purpose.

Behavioral analysis for a transitory gathering of people: Onnela [16] studies population dynamics for large transitory gatherings using messaging and call data records. Since many of these people are also expected to carry mobile devices, VECTORS can help provide a video feed for better contextual analysis of the events and people behavior. VECTORS can also be used for improving law enforcement in similar scenarios.

Rural deployments: We intend to use VECTORS in agricultural environments to help track cattle in a field using cameras mounted on poles or trees with a solar-powered video capture device. Furthermore, we aim to use it for monitoring the activities like the harvesting of fruits, flowers, and crops. Video monitoring can discourage theft and reduce the need for the workforce to guard many of the farms and orchards.

5. Limitations and future work

The current implementation of VECTORS uses Linux computers at the source and destination for processing video. The connection of Android devices to these computers via ADB complicates the deployment of VECTORS. Moreover, the SHM encoder is not parallelized and it takes a lot of time to encode high-resolution video. The present SVC adaptation only utilizes additive-increase/multiplicative-decrease. More advanced adaptation algorithms, including machine learning approaches, are open topics for research.

In future iterations of this project, we plan to port SHM to Android for a simpler setup of VECTORS. We also plan to improve the interoperability of VECTORS by implementing the Bundle protocol [17]. Lastly, we intend to experiment with alternate opportunistic routing protocols. Other possible extensions to VECTORS include the enforcing of a storage quota on each device and the addition of security in communication.

6. Conclusions

We were able to successfully have a media flow for the two-week interval with delays in playback varying between a couple of hours to a day. We had fifteen active participants, which is representative of rural deployments. The initial experiment shows that in the true rural scenario, VECTORS can be effective for communicating such content since these users will not have infrastructure-based Wi-Fi.

Conflict of interest

The authors declare that they have no conflict of interest.

Appendix

The contact traces and simulation settings are publicly made available at https://github.com/swifiic/the-one.

References

[1] Wilk S, Kopf S, Effelsberg W. Video composition by the crowd: a system to compose user-generated videos in near real-time. In: Proceedings of the 6th ACM multimedia systems conference. ACM; 2015, p. 13–24.

[2] Cerf V, Burleigh S, Hooke A, Torgerson L, Durst R, Scott K, Fall K, Weiss H. Delay-tolerant networking architecture (No. RFC 4838); 2007.

[3] Unanue I, Urteaga I, Husemann R, Del Ser J, Roesler V, Rodríguez A, Sánchez P. A tutorial on H 264/SVC scalable video coding and its tradeoff between quality, coding efficiency and performance. In: Recent advances on video coding. InTech; 2011.

[4] Boyce JM, Ye Y, Chen J, Ramasubramonian AK. Overview of SHVC: Scalable extensions of the high efficiency video coding standard. IEEE Trans Circuits Syst Video Technol 2016;26(1):20–34.

[5] Mitchener W, Vadhat A. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, 2000.

[6] Spyropoulos T, Psounis K, Raghavendra CS. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking. ACM; 2005, p. 252–9.

[7] Lenas SA, Burleigh SC, Tsaoussidis V. Bundle streaming service: design, implementation and performance evaluation. Trans Emerg Telecommun Technol 2015;26(5):905–17.

[8] Morgenroth J, Pögel T, Wolf L. Live-streaming in delay tolerant networks. In: Proceedings of the 6th ACM workshop on challenged networks. ACM; 2011, p. 67–8.

[9] Blanchet M. Postellation: an enhanced delay-tolerant network (dtn) implementation with video streaming and automated network attachment. In: SpaceOps 2012; 2012. p. 1279621.

[10] Schildt S, Morgenroth J, Pöttner WB, Wolf L. IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation. Electron Commun Eur Assoc Softw Sci Technol 2011;37.

[11] Camps-Mur D, Garcia-Saavedra A, Serrano P. Device-to-device communications with Wi-Fi Direct: overview and experimentation. IEEE Wireless Commun 2013;20(3):96–104.

[12] Nayak R. Announcing Nearby Connections 20: fully offline, high bandwidth peer to peer device communication; 2017. https://android-developers.googleblog.com/2017/07/announcing-nearby-connections-20-fully.html. [Last updated 31st 2017, last accessed 17th 2018].

[13] SHM reference software 12.4. https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/tags/SHM-124/. [Last accessed 17th 2018].

[14] Keränen A, Ott J, Kärkkäinen T. The ONE simulator for DTN protocol evaluation. In: Proceedings of the 2nd international conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering); 2009, p. 55.

[15] Juang P, Oki H, Wang Y, Martonosi M, Peh LS, Rubenstein D. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. ACM SIGARCH Comput Archit News 2002;30(5):96–107.

[16] Onnela JP, Khanna T. Investigating population dynamics of the Kumbh Mela through the lens of cell phone data; 2015. arXiv preprint arXiv:1505.06360.

[17] Scott K, Burleigh S. Bundle protocol specification (No. RFC 5050); 2007.