Fountain-code Aided File Transfer in Vehicular Delay Tolerant Networks

Seyed Masoud Mousavi LANGARI[1], Saleh YOUSEFI[2], Sam JABBEHDARI[1]
[1]*Department of Computer Engineering, North Tehran Branch, Islamic Azad University, 1667934783 Tehran, Iran*
[2]*Department of Computer Engineering, Faculty of Engineering, Urmia University, Urmia 15311-57561, Iran*
sm_mousavi@iau-tnb.ac.ir, s.yousefi@urmia.ac.ir, s_jabbehdari@iau-tnb.ac.ir

***Abstract*—We propose a mechanism for facilitating file transferring in Vehicular Delay Tolerant Networks. The proposed architecture includes using Fountain coding in the application layer, UDP in the transport layer and a proposed DTN routing algorithm in the network layer. It is assumed that files are coded based on a sample of Fountain codes which does not need in-order reception of packets. As a result, there is no need of using close-loop reliable protocols such as TCP, hence suffering from their different overheads; as a result, UDP can be used in the transport layer. In the network layer, we propose a novel DTN routing algorithm based on AODV and Store-Carry and Forward policy. This algorithm (named as AODV-DTN) uses a cross layer interaction between the network and the application layer. Results of extensive simulations study for highway scenarios show that the proposed architecture leads to a better performance in terms of file delivery ratio and byte throughput when compared with FOUNTAIN and classic FTP scenarios. Furthermore, the negative effect of increasing file size is mitigated in comparison to other alternatives. It is also shown that for delay tolerant and long-distanced inter-RSU communications the proposed architecture behaves sufficiently well.**

***Index Terms*—Ad hoc Network, Buffer storage, Disruption tolerant network, Error correction codes, Routing protocols.**

I. INTRODUCTION

In a wireless ad hoc network, an opportunistic routing strategy is a strategy in which there is no predefined rule for choosing the next node in the forwarding process of a message. A popular sample application for such a routing policy is in networks suffering from intermittent connectivity, i.e., end-to-end communication paths are not available continuously between sources and destinations.

An emerging class of ad hoc networks, called VANETs exploits transportation systems in order to transfer data. In these networks, vehicles act as mobile nodes used for carrying data among vehicles and /or fixed RSUs. Vehicles can exchange data messages, have the capability to download, store, and upload the data messages from/to the other vehicles as well as road side infrastructure. Due to movement of vehicles, VANETs always suffer from frequent disconnections. Consequently, traditional routing protocols such as AODV [1] cannot be directly applied to these networks. The reason for this inapplicability could be ascribed to limited buffer and dropping the packets in such condition. As a result, it cannot address the requirements of DTN (e.g., frequent disconnections). However, there are many applications which tolerate delay as long as delivery is guaranteed. The networks (protocols) which are customized for such applications are generally called DTN. To send data to a destination in a DTN, one obvious solution is S-CF strategy. In this paradigm, when a node receives a message but it is not connected to any other neighboring nodes, it stores the messages and carries them until an appropriate communication opportunity arises (one or more mobile nodes are appeared); thereafter, it will forward the messages to the new neighbor (s).

The main application we consider is file transferring. Due to their nature, these applications need in-order and fully reliable (from the application's viewpoint) packet reception. As a result, TCP is normally used as the transport layer protocol in which ACK packets are exchanged to guarantee reliable and in-order reception of packets. However, due to intermittent connectivity in mobile environments such as VDTNs, this paradigm which basically addresses fixed network suffers from several shortcomings. In fact, imposing high restrictions on the reception order of packets and acknowledging their reception, reduces the chance of successful packet reception and increases delay and loss rate. To enhance the performance of message exchange in VDTN, in this paper we have thus proposed the use of fountain coding [2] in the application layer along with UDP in the transport layer.

Our contributions in this paper are as follows:

(a) Proposing a new architecture for facilitating DTN in VANETs which includes using Fountain coding, UDP and a DTN routing policy. The need to in-order file reception is obviated by exploiting Fountain coding in the application layer; receiving enough number of distinctive packets is sufficient. In other words, for recovering the file at the destination, all packets have the same value and their reception order is not important. This also obviates the need for using TCP in the transport layer and UDP suffices.

(b) Solving the buffer space problem in the network layer and as the DTN routing policy requires, we propose AODV-DTN algorithm which adds S-CF policy to the basic AODV algorithm. The proposed architecture hinders buffered packets from being dropped; in fact, it sends them to the application layer where a large amount of buffer is provided. In other words, a sort of cross layer interactions (between the network and the application layer) is used for transmitting packets.

(c) Discussing performance trade-offs: (i) byte throughput vs. file size (ii) delivery ratio vs. file size, (iii) download time vs. file size, , (iv) inter-RSU byte throughput vs. file size. Using those, we examine how file size affect the

Digital Object Identifier 10.4316/AECE.2013.04020

efficiency of our architecture. In order to evaluate the performance of proposed architecture we have conducted extensive simulation study.

The reminder of this paper is organized as follows. In section 2, we review the related works. In section 3, we explain the fountain code and its characteristics. In section 4, we describe in details the proposed VDTN approach. In section 5, we bring the results of simulation of the proposed approach. Finally, the paper is concluded in section 6.

II. Related Works

Various routing protocols have been introduced and designed for DTNs. In following, we categorize the most important ones. The first category is based on having some prior knowledge about nodes' mobility pattern [3-4]. The probabilistic routing was used based on history of nodes' encounters and transitivity [5]. Hence, a message was passed to a new relay only if it had higher delivery predictability. It should be noted that in most practical VANETs scenarios, the schedules of encounters may not be predictable. Even if the schedules are known, there may be some errors. In the worst case, if the movements of the nodes are random, no assumptions can be made about their movement. In this paper, we neither use prior information nor do we control vehicles` movement patterns for file transfer.

In the second category, nodes are aware of their location through some methods such as GPS. IHLAR combined geographic with topology based routing to reduce the end-to-end delay [6]. Hello packets were used to calculate the delivery probability [7]; furthermore, Hello packets were deployed to propose ORION improving delay and delivery ratio [8]. PRNFP proposed a position based routing algorithm which enhanced the performance of the system [9]. Digital map, connectivity graph, location of destination vehicle, and the duration of connectivity among vehicles were deployed to achieve a stable route and reduced network load [10]. Some of these methods try to send message based on local information in each step that may cause to create a loop because each node selects the closest node to destination from its local point of view. Therefore, such methods may suffer from local phenomena. Moreover, greedy forwarding may lead to a dead end which means there is no closer neighbor to the destination.

The third category of previous works is based on exploiting relay nodes. The message delivery probability was evaluated in two-hop relay with erasure coding [11]. In [12] some stationary devices were located at crossroads where vehicles met them. The mentioned stationary devices improve packets delivery probability by practicing store and forward policy. By using these devices, authors of [13] tried to replicate data for maintaining Distributed Data Base (DDB). However, the weakness of these methods is placement of the stations. The optimal placement was shown to be NP-Hard [14].

One of the simplest and earliest methods in DTN is replication methods. A simple routing protocol named Epidemic Routing was proposed based on S-FC and replication [15]. Inspiring from this basic method, other authors try to propose more intelligent methods, aiming at limiting the number of packets propagated in network. The average copy count per message was tried to be reduced in different time steps called periods [16]. In each period, some additional copies were sprayed into the network, and each period had its own waiting time for message delivery. The Spray&Wait [17] algorithm sent packets to a certain number of vehicles which were selected randomly, and waited until one of these vehicles met the destination. In the wait phase, aiming at reaching a better performance, took advantage of mobility information [18]. However, the clear shortcoming of replication-based schemes is waste of bandwidth.

Different authors combined network coding with other DTN methods. The effect of different spraying algorithms and cost reduction of erasure coding were studied [19]. In [20], the authors proposed an idea in area of network coding to improve data delivery and reduce the number of transmitted bytes by flooding. In [21], the authors implemented the epidemic routing with network coding, and to obtain better performance they introduced adaptive scheduling mechanism to ensure that the buffer of nodes is enough for current amount of data. The performance of epidemic routing in presence of network coding in opportunistic networks was compared with the sole use of replication [22]. In [23], by using erasure coding, along with estimation based routing schemes, lower delivery delay and faster distributing of message blocks were reported. In [24], the goal was to study a class of replication methods that included coding to enhance the probability of successful delivery within a given time limit.

III. Fountain Codes

A special type of network coding is a Fountain Codes [25], also known as rateless erasure codes. Fountain code does not require a feedback packet for acknowledgment and the original source file can be rebuilt up from any subset of encoding symbols from the given set which is equal or only slightly larger than the source file. Raptor Codes [26], Luby Transform Codes [27], Online Codes [28], and different kinds of fountain codes [29-30] are some examples of this coding.

Fountain codes constitute a class of rateless codes, which can generate an infinite number of encoded packets based on the source file. For encoding packets, at first, a degree distribution should be chosen which determines the number of symbols that would be summed up together, using XOR operation on the bit level, into one output symbol. Each encoded symbol is generated randomly and independently by using the aforementioned degree distribution. The key factor of encoding and decoding process is degree distribution since the efficiency of the codes strictly depends on this issue; therefore, some authors try to design an optimized degree distribution to improve the performance of fountain codes [31-33]. Through reverse XOR operation, the decoding process is done at the receiver with respect to the degree distribution in the header of the received packet. The most important characteristic of fountain codes is that the source data packets can be recovered from any subset of the received packets.

Generally, $N = K(1 + \varepsilon)$ packets are needed to decode the original file successfully where ε is the overhead and K is the number of original packets that are going to be

transmitted. The overhead, so-called ε, normally is variant between 5 to 100% which depends on the implementation [34]. If $N < K$, the receiver does not get enough encoded packets to re-build the original file. If $N = K$, it is conceivable that receive is able to recover the original file. And, finally, if $N = K + E$ and $N > K$ where E is supplementary packets, the probability of recovering the original file at the destination is $(1 - \delta)$ and δ is upper-bounded by $2^{-K\varepsilon}$. As a result, the larger the file size is, the higher the probability of receiving successfully at receiver becomes. The encoding and decoding process of fountain code has a low complexity and the required space for storing these processes are linear; therefore, the only cost of this coding is related to its excess number of encoded packets in term of load of the network. These characteristics of fountain code demonstrate that not only it can be efficient and reliable as TCP, but also it is universal and tolerable in intermittent connectivity environments such as VDTNs.

Fountain code in application layer for encoding and decoding process and PUMA in network layer were used to reduce the delivery time and bandwidth occupation [35]. Fountain codes were used to improve file transfer probability and efficiency [36]. A comparison of fountain code and TCP in case of file transfer showed that fountain code outperforms TCP in typical cases of operation [37]. CFP [38] integrated optimal probabilistic forwarding scheme with fountain codes to reach better delivery ratio while the waste of the resources was avoided. FOCAR was proposed to obtain high reliability and low end-to-end delay [39]. In addition, it was shown that Fountain coding could improve the data delivery in VANETs compared to other traditional protocols [2]. In this paper, we are seeking a thorough and systematic understanding of the benefits and performance gains when Fountain coding is used in VANETs along with S-CF policy. Our approach uses neither a priori knowledge of network connectivity nor any control over nodes' mobility. To the best of our knowledge, this sort of investigation has not been performed so far. The main advantages of fountain codes lay on its positive characteristics can be summarized in:

1- Low complexity of encoding and decoding
2- Reconstructing the original file from any subset
3- Receiving packets out-of-order at the destination
4- Obviating the demand of an acknowledgement for receiving a packet at a destination.

IV. Proposed Architecture

As illustrated in Fig. 1, our proposed architecture for file transferring includes using Fountain coding in the application layer, UDP in the transport layer and a cross layer mechanism which extends AODV for DTN (named as AODV-DTN). We assume file chunks are encoded by the sender according to a sample of fountain codes (such as Raptor [26]) while the sender is off-line and stores packets in its own memory. Since fountain coding is used in the application layer, the order of packet reception is not important and all coded packets have equivalent values. This feature satiates the demand of using TCP and its ACK mechanism in the transport layer; hence, deploying a simple UDP along fountain coding is sufficient. The main logic behind choosing this architecture is that in intermittent-connective environments such as VDTN, data packets and even ACKs may be in risk of loss due to frequent disconnections leading to several retransmissions. In such environments, traditional routings like AODV do not appear to act well. Indeed, they cannot store many packets for a long time due to lack of buffer space; consequently, data packets are dropped from buffer after a short time. In addition to using fountain coding in the application layer, to find a better chance to deliver data from sources to destinations, there is a cross-layer interaction between AODV-DTN and the application layer; that is, we add a large FIFO buffer to the application layer (Fig. 1). *Link_Breakage* and *Neighbor_Found* signals are deployed in our paradigm sent from AODV-DTN toward the application layer. Each time a link breakage is detected, AODV-DTN sends a *Link_Breakage* signal to the application layer to inform itself that destination is changed temporarily; thereafter, AODV-DTN instead of buffering the packets itself sends the packets toward the FIFO module in the application layer for the sake of buffering.

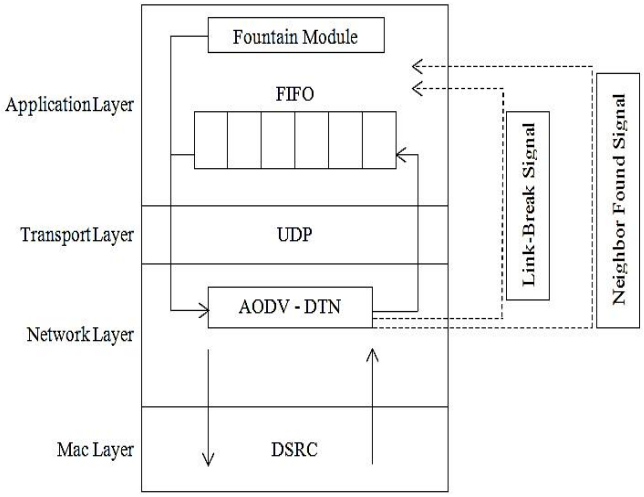

Figure 1. Proposed Architecture

The large application layer buffer prevents packet loss and dropping at the network layer. Since the application is tolerable, we assume that the TTL of packets are large enough to ensure that all packets are delivered to the destination; indeed, there is no packet loss due to TTL expiry. Later, if a *temporary destination* (carrier vehicle) detects a new neighbor (through HELLO mechanism; see the next paragraph) that have a route toward the *primary destination*, the network layer gives *Neighbor_Found* signal to the application layer. Thus, buffered data will be sent to the newly found neighbor and then toward the *primary destination*. In the following, we explain the process in more details.

In RFC [3561], the HELLO mechanism is proposed in order to determine network connectivity and offer connectivity information for possible available neighbors through reception of broadcast control message. Each vehicle by broadcasting a HELLO message every in *HELLO_INTERVAL* [1] can assist in identifying whether a neighbor has joined or left the network. Each vehicle maintains a neighbor table to store information of vehicles

within its transmission range, as well as a route table to store information about routes to destinations. In our new architecture, so-called HELLO messages are utilized to detect breakage and establishment of links.

Data are sent through multi-hop communication from a source vehicle to its related destination, and the active route for file transferring never expires unless a disconnection occurs. When the *primary destination* is unavailable meaning that a disconnection happens and an interruption is detected by an intermediate vehicle; hence, a *Link_ Breakage* signal is sent to the application layer of this intermediate vehicle which is named *temporary destination*. As shown in Fig. 2, from now on packets are sent to this *temporary destination,* and the AODV-DTN sends received packets to the application layer for buffering.

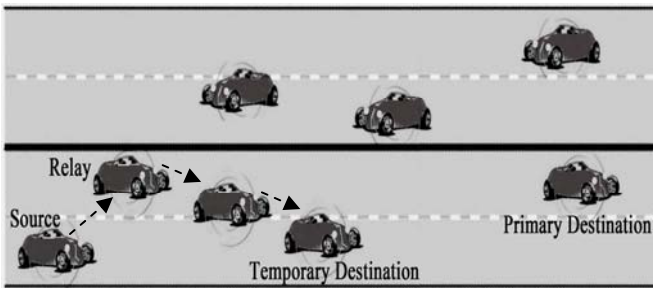

Figure 2. Implemented Store-Carry and Forward policy

During the routing process, if the connection between source and the current *temporary destination* is interrupted, the last available vehicle becomes another *temporary destination*. As we assume packets do not expire because of TTL, they can be carried until an opportunity raises for delivering to the *primary destination*. Whenever any of the *temporary destinations* finds a new neighbor, the AODV-DTN protocol checks whether there is a route to the *primary destination* through the new neighbor by using route discovery [1]. If so, AODV-DTN sends *Neighbor_Found* signal to the application layer through cross layer mechanism (see Fig. 1). By receiving this signal in application layer, the *temporary destination* commences forwarding its buffered packets. Otherwise, it carries the buffered packets until finding a neighbor that has got a path to the *primary destination*. In forwarding phase, two cases are raised:

- If the desired file is not fully received in a temporary destination, it only tries to forward its buffered file chunks.
- If the original file is received completely in a *temporary destination*, the original file can be rebuilt up at this vehicle. Afterwards, it can act as a source vehicle and generate encoded symbols of its own from the full content to the *primary destination*. This issue helps the *primary destination* to download and receive the desired file faster and more reliable.

While file chunks are being forwarded, if a route breaks, last available vehicle becomes a *temporary destination* for correspondence source which previously have been a *temporary destination*. Accordingly, this process continues until data are delivered to the *primary destination*. The operational procedure is shown in Fig. 3. In other words, we may have more than one *temporary destination* each responsible for partial file delivery to the *primary destination*. As we use the fountain coding in the application layer, packets do not need to be received in-order in the *primary destination*. Indeed, it does not matter which *temporary destination* delivers its packets first to the *primary destination*, and the sequence of received packets in the *primary destination* is not important. Furthermore, file chunks can be downloaded in parallel from a variety of *temporary destinations*. The *primary destination*, after downloading the desired file thoroughly, can close all connection without concerning packets where come from and duplication of them [40]. This characteristic of fountain-coded data is critical in our proposed architecture so that we are able to use multiple *temporary destinations* trying to send their data toward the *primary destination* in an independent manner.

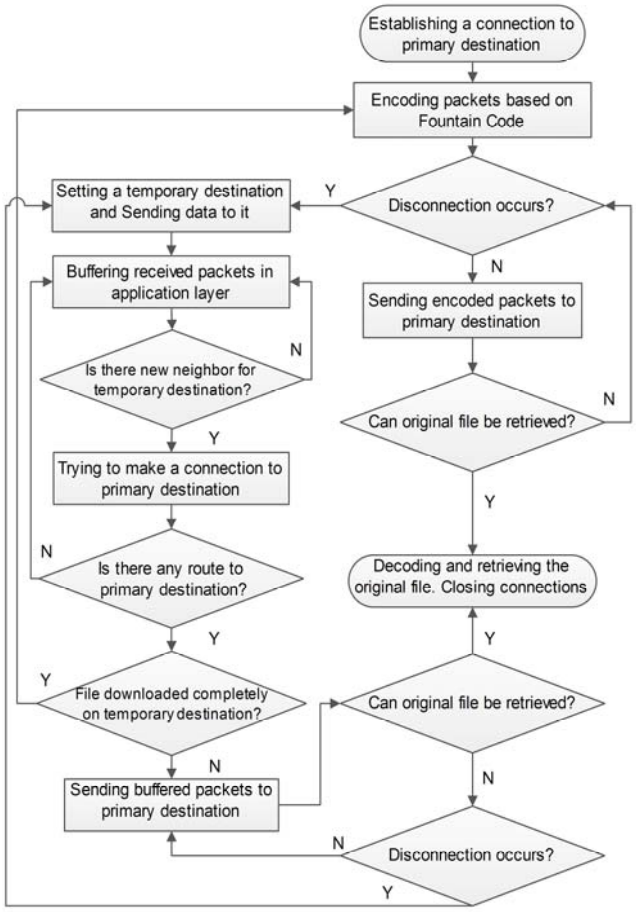

Figure 3. Operational procedure

V. Simulation

In this section, we show that fountain coding and S-CF policy results in significantly improved performance in terms of byte throughput and delivery ratio. We implemented all parts of the proposed approach in GloMoSim2.03 library-simulator [41]. The MAC layer is IEEE 802.11 (the base of DSRC standard) and the transmission range of vehicles is 250 m. The bandwidth is set to 6 Mbps and we use SUMO [42] as mobility generator.

The results of this paper are for a highway scenario with 4 lanes in each direction and 30 km in length. The speed

levels are 80, 100, and 120 with a probability of occurrence of 25%, 50%, and 25% respectively. Furthermore, we use 30 minutes warm up to reach a normal distribution of vehicles and 600 seconds for simulation time. It is assumed that the original file is recovered by an overhead of 10% (note that current fountain codes are able to achieve the overhead of 5% easily). The implementation of coding and decoding algorithm for a special type of fountain codes is beyond the scope of this paper. Hence, in the implemented scenario, the sender vehicle sends out a sequence of packets to the receiver. Whenever the receiver gets 110% of original file packets (distinctive packets), no matter whether the order is preserved or not, it sends back an ACK to the sender; thereafter, sender stops sending out new packets preventing imposing extra overhead to the network. Since vehicles normally tend to start communication with neighboring vehicles, in the conducted simulation, we choose 30 vehicle pairs such that their hop count distance is around 7 or 8 at the time of connection establishment. We simulate different scenarios with these 30 selected pairs, and for each pair, we repeat the simulation 10 times. The depicted results in following figures are the average among of these independent simulations. The following scenarios are considered for evaluation the proposed architecture.

• FTP: in which each vehicle sends out file chunks of size 1KB using TCP. In the application layer, a standard implementation of FTP protocol is used. Besides, standard version of AODV [1] is used as the routing protocol.

• FOUNTAIN: in which each vehicle sends out file chunks of 1KB using UDP. In application layer, coding and decoding based on a sample of fountain codes like Raptor [26], LT [27] is utilized, and in the network layer, standard AODV is deployed. This scenario was proposed in [2].

• FOUNTAIN S-CF: in which fountain coding is used in application layer and the proposed AODV-DTN algorithm is used as the routing protocol. Details of the architecture of this scenario have been explained in section 4.

Byte throughput is an important metric which should be evaluated in VDTNs because there might be the necessity to provide a kind of resume facility if a vehicle is not able to finish file(s) download in one-step. One example could be file download from multiple Road Side Units (RSU). To determine the effect of FIFO queue size on byte throughput in our proposed architecture, we measure byte throughput of the FOUNTAIN S-CF when the FIFO queue size is limited by different values (from 1MB to 10 MB). Fig. 4 shows that when file size is small and FIFO queue size is maximized, files have more chance to be thoroughly delivered. However, as can be seen, when files enlarge while FIFO queue size is kept small, the byte throughput is low. That is, several packets are dropped. This issue is related to lack of space in buffer; in other words, if there is no free space in buffer and the connection between a source and *temporary destination* is still available, packets will be dropped, hence contributing to low byte throughput. However, for alleviating the problem one can implement the buffer on disk instead of RAM. Since the application is delay tolerant then the imposed delay of retrieving file chunks from disk can be tolerable. On the other side, a proper prefetching of file chunks can decrease the imposed I/O delay noticeably.

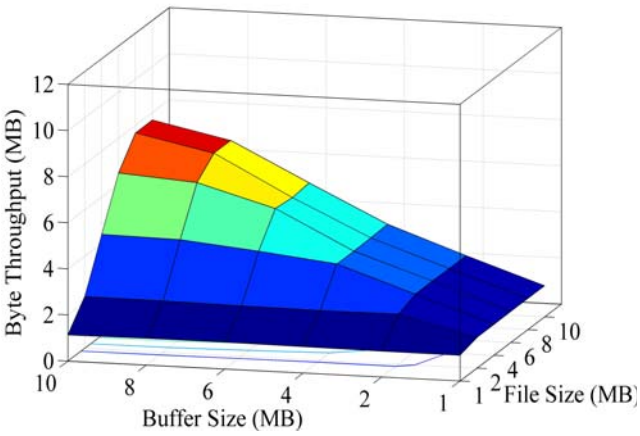

Figure 4. Byte throughput of FOUNTAIN S-CF with different buffer size

In Fig. 5, the byte throughputs of different protocols are compared considering no limitation on the buffer size. As can be seen, the FOUNTAIN S-CF's throughput in terms of byte count is higher than that of FOUNTAIN and FTP. By taking advantage of fountain codes and S-CF policy, we can transfer a larger number of bytes in comparison to other scenarios. Since FTP should send packets in both direction (i.e., data packets in one direction and ACK in reverse direction), it causes collision and increment in load of the network; therefore, byte throughput will be diminished. In contrast, FOUNTAIN S-CF sends only data packets, without ACKs; furthermore, the reason of this result is ascribed to equivalent value of file chunks and reconstruction of the original file from any subsequence of file chunks.

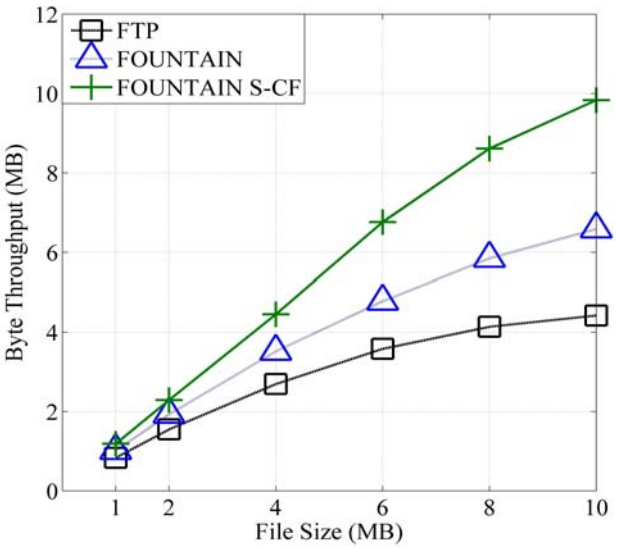

Figure 5. Byte throughput comparison

Another important metric for comfort applications which should be analyzed is file delivery ratio; because regarding their nature, vehicles can make use of such applications only if the related files are downloaded completely. The plot of delivery ratio in Fig. 6 shows FOUNTAIN S-CF has a higher delivery ratio in comparison to other protocols. An ordinary AODV has a limited buffer and cannot store many packets in the network layer for a long time; indeed, if a route is not established in a timely manner, all packets should be dropped from the buffer. This behavior has impact

on delivery ratio since fewer packets are delivered in FTP and FOUNTAIN scenario.

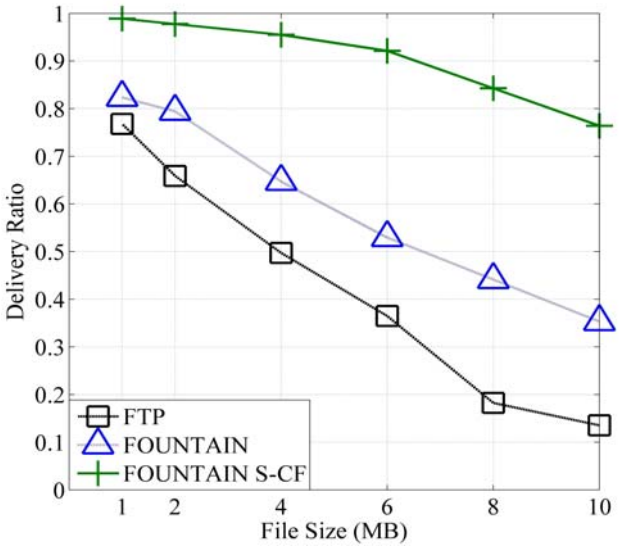

Figure 6. File delivery ratio comparison

Fountain codes increase the possibility of recovering files from out-of-ordered file chunks. The result of FOUNTAIN reflects this setting previously investigated in [2]. This feature can fade and cover the inefficiency of AODV in VDTN; therefore, as can be seen in Fig. 6, FOUNTAIN outperforms FTP. With respect to fountain coding characteristic, vehicles can gather file chunks and pursue their incomplete download form any vehicle they meet, in any contact opportunity. As followed from Fig. 6, FOUNTAIN S-CF scenario shows the best performance. In fact, due to the existence of AODV-DTN algorithm in the network layer, reception probability of file chunks is improved. As mentioned in section 4, in the proposed architecture, there might be several *temporary destinations* each of which may have and carry different file chunks. As a result, the chance of a complete file reception at the *primary destination* is raised. As Fig. 6 illustrates, FOUNTAIN S-CF scenario delivered more files compared to other protocols when the file size increased. In other word, the negative effect of file size on file delivery ratio in the FOUNTAIN S-CF is less than other scenarios.

We compare download time in above-mentioned scenarios and the results are obtained based on those files that are received completely at the destination. As shown in the Fig. 7, FOUNTAIN S-CF requires more time to download files completely in comparison with FTP and FOUNTAIN. However, as observed in Fig. 6, file delivery ratio is higher in FOUNTAIN S-CF scenario. It is actually expected that using AODV-DTN impose some extra delay due to carrying packets until finding an opportunity to deliver buffered packets to primary destination. Therefore, in those applications where delay is not a critical concern, FOUNTAIN S-CF can be a good candidate for file exchange.

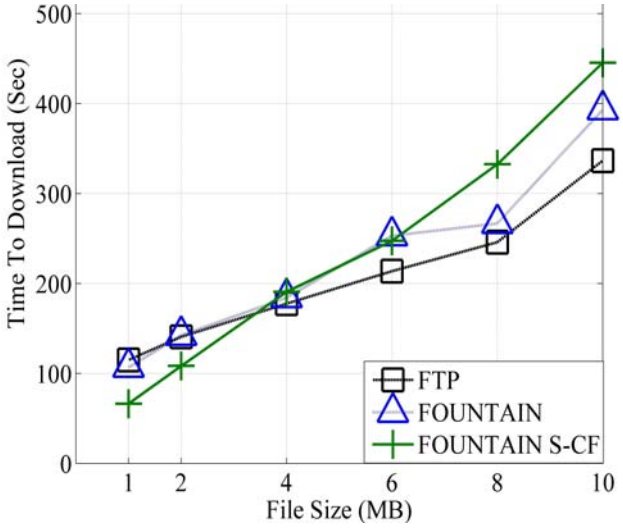

Figure 7. Time to download comparison

Fig. 8, 9, and 10 evaluate using the proposed architecture when stationary (fixed) Road Side Units (RSUs) intend to communicate with each other. This can be useful in many applications (including comfort and life safety) when some delay tolerant data are going to be exchanged between RSUs in highways without the existence of infrastructure. We set an experiment in which two RSUs with different distances (from 1KM to 6KM) in a highway are considered. Data are routed (or carried) using solely vehicles. When the distance set to 1KM, the destination RSU is able to receive files with different sizes thoroughly in FOUNTAIN S-CF and FOUNTAN scenario. In contrast, FTP has a low byte throughput because of its characteristics (i.e., need to backchannel for acknowledgment). Generally, the overall byte throughput may degrade as the distance between RSUs increase, as it adds more path loss. However, the increment of distance does not have a negative effect on FOUNTAIN S-CF; it is due to using AODV-DTN algorithm which practices S-CF policy. Data are sent by the source RSU, and those vehicles which are moving along the source RSU start to carry the packets to the destination RSU. When the carrier vehicles pass by the destination RSU, they start to forward their buffered packets to the destination.

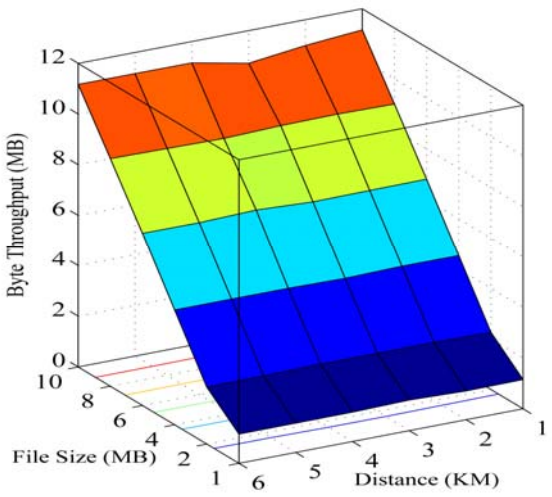

Figure 8. Effect of distance on FOUNTAIN S-CF`s byte throughput

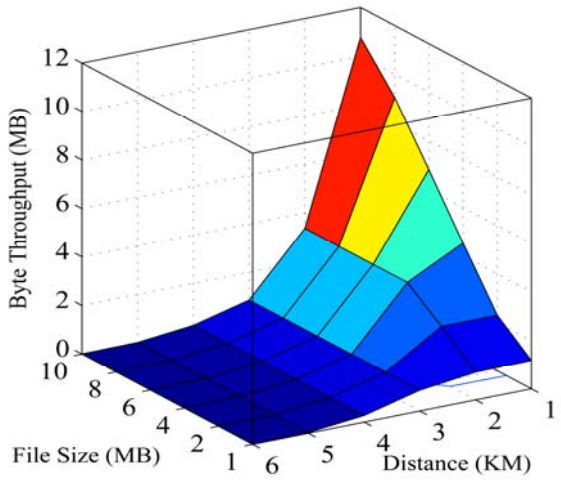

Figure 9. Effect of distance on FOUNTAIN`s byte throughput

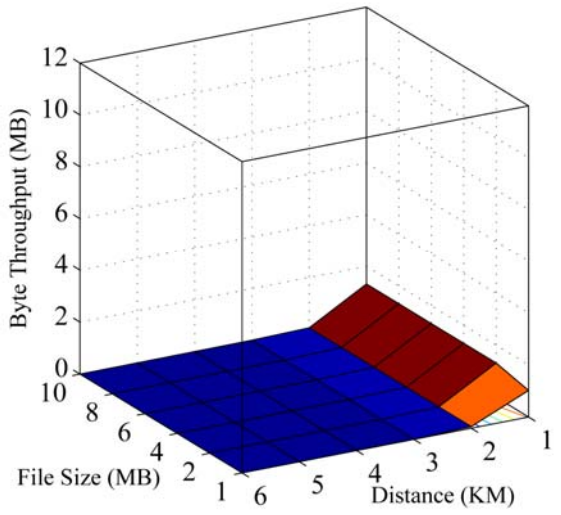

Figure 10. Effect of distance on FTP`s byte throughput

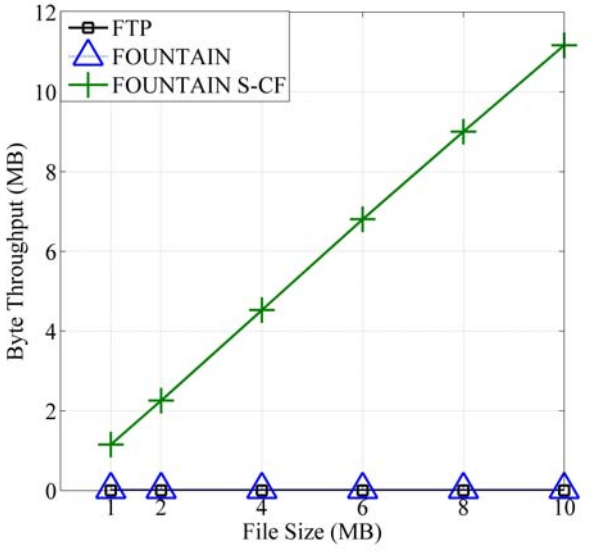

Figure 11. Inter-RSU throughput for different scenarios

Fig. 11 illustrates the comparison of mentioned protocols when two RSUs are located 6KM far from each other. As shown in Fig. 11, the distance of 6KM is a too long in such a way that even in FOUNTAIN scenario the destination RSU does not receive any of the transmitted packets. Furthermore, FTP needs a stable connection which cannot be provided because of fixed position of RSUs and mobility of vehicles. However, in FONTAIN S-CF scenario all packets can be carried and transmitted successfully. In Fig. 11, the byte throughput results are reported and as one can see, the byte throughput is 110% of file sizes. This difference is due to the 10% overhead which is imposed because of the application of fountain codes in the application layer.

VI. Conclusion

In this paper, we proposed architecture for file transferring in VDTN. The proposed architecture included using fountain coding in the application along with UDP in the transport layer. We also proposed a novel DTN routing algorithm based on the well-known AODV named as AODV-DTN and used it as routing protocol in the proposed architecture. The proposed approach achieves a higher throughput along with a better reliability compared to other approaches suggested thus far in the literature. Our results depicted that, the delivery ratios are higher in the proposed architecture compared to other alternative scenarios. Furthermore, the negative effect of increasing file size on file delivery ratio is lower in the proposed architecture. On the other hand, download time increases due to S-CF policy compared to other scenarios. We also showed that for long-distanced delay tolerant inter-RSU communications, the proposed architecture shows a very good performance despite the failure of other alternative approaches. As a result, our proposed architecture can be a good candidate for delay tolerant file transferring in networks which suffer from intermittent connectivity including VANETs. In future works, we intend to control the TTL of the messages.

Acknowledgment

We want to express our sincere gratitude to Mr. Keywan Zayer for his collaboration in preparing this article.

References

[1] C. E. Perkins and E. M. Royer, “Ad-Hoc on-Demand Distance Vector Routing,” in Second IEEE Workshop on Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA ’99, 1999, pp. 90–100.

[2] S. Yousefi, T. Chahed, S. M. M. Langari, and K. Zayer, “Comfort Applications in Vehicular Ad Hoc Networks Based on Fountain Coding,” in Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st, 2010, pp. 1–5.

[3] G. Xue, Y. Luo, J. Yu, and M. Li, “A Novel Vehicular Location Prediction Based on Mobility Patterns for Routing in Urban Vanet,” EURASIP Journal on Wireless Communications and Networking, vol. 2012, no. 1, p. 222, Jul. 2012.

[4] Q. Yuan, I. Cardei, and J. Wu, “An Efficient Prediction-Based Routing in Disruption-Tolerant Networks,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 1, pp. 19–31, 2012.

[5] F. Dang, X. Yang, and K. Long, “Spray and forward: Efficient routing based on the Markov location prediction model for DTNs,” Sci. China Inf. Sci., vol. 55, no. 2, pp. 433–440, Feb. 2012.

[6] D. Luo and J. Zhou, “An Improved Hybrid Location-Based Routing Protocol for Ad Hoc Networks,” in 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2011, pp. 1–4.

[7] Y. Feng, H. Gong, M. Fan, M. Liu, and X. Wang, “A Distance-Aware Replica Adaptive Data Gathering Protocol for Delay Tolerant Mobile

Sensor Networks," Sensors, vol. 11, no. 12, pp. 4104–4117, Apr. 2011.

[8] S. Medjiah and T. Ahmed, "Orion Routing Protocol for Delay Tolerant Networks," in 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1–6.

[9] H. Idjmayyel, B. R. Qazi, and J. M. H. Elmirghani, "A Geographic Based Routing Scheme for Vanets," in Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On, 2010, pp. 1–5.

[10] M. Ramakrishna, "Dbr-Ls: Distance Based Routing Protocol Using Location Service for Vanets," in 2011 Annual IEEE India Conference (INDICON), 2011, pp. 1–4.

[11] J. Liu, X. Jiang, H. Nishiyama, and N. Kato, "On the Delivery Probability of Two-Hop Relay MANETs with Erasure Coding," IEEE Transactions on Communications, vol. 61, no. 4, pp. 1314–1326, 2013.

[12] V. N. G. J. Soares, F. Farahmand, and J. J. P. C. Rodrigues, "Improving Vehicular Delay-Tolerant Network Performance with Relay Nodes," in Next Generation Internet Networks, 2009. NGI '09, 2009, pp. 1–5.

[13] A. Lieskovsky, J. Janech, and T. Baca, "Data Replication in Distributed Database Systems in Vanet Environment," in 2011 IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS), 2011, pp. 304–307.

[14] F. Farahmand, I. Cerutti, A. N. Patel, J. P. Jue, and J. J. P. C. Rodrigues, "Performance of Vehicular Delay-Tolerant Networks with Relay Nodes," Wireless Communications and Mobile Computing, vol. 11, no. 7, pp. 929–938, 2011.

[15] A. Vahdat, and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[16] E. Bulut, Z. Wang, and B. K. Szymanski, "Cost-Effective Multiperiod Spraying for Routing in Delay-Tolerant Networks," IEEE/ACM Transactions on Networking, vol. 18, no. 5, pp. 1530–1543, 2010.

[17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, New York, NY, USA, 2005, pp. 252–259.

[18] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility," in Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07, 2007, pp. 79–85.

[19] E. Bulut, Z. Wang, and B. K. Szymanski, "Cost Efficient Erasure Coding Based Routing in Delay Tolerant Networks," in 2010 IEEE International Conference on Communications (ICC), 2010, pp. 1–5.

[20] J. Widmer and J.-Y. Le Boudec, "Network Coding for Efficient Communication in Extreme Networks," in Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, New York, NY, USA, 2005, pp. 284–291.

[21] Q. Zhang, Z. Jin, Z. Zhang, and Y. Shu, "Network Coding for Applications in the Delay Tolerant Network (DTN)," in 5th International Conference on Mobile Ad-hoc and Sensor Networks, 2009. MSN '09, 2009, pp. 376–380.

[22] Y. Lin, B. Li, and B. Liang, "Stochastic Analysis of Network Coding in Epidemic Routing," IEEE Journal on Selected Areas in Communications, vol. 26, no. 5, pp. 794–808, 2008.

[23] Y. Liao, K. Tan, Z. Zhang, and L. Gao, "Estimation Based Erasure-Coding Routing in Delay Tolerant Networks," in Proceedings of the 2006 international conference on Wireless communications and mobile computing, New York, NY, USA, 2006, pp. 557–562.

[24] E. Altman and F. De Pellegrini, "Forward Correction and Fountain Codes in Delay-Tolerant Networks," IEEE/ACM Transactions on Networking, vol. 19, no. 1, pp. 1–13, 2011.

[25] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, New York, NY, USA, 1998, pp. 56–67.

[26] A. Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, no. 6, pp. 2551–2567, 2006.

[27] M. Luby, "LT codes," in The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings, 2002, pp. 271–280.

[28] P. Maymounkov, "Online codes," Technical report, New York University, 2002.

[29] M. Asteris and A. G. Dimakis, "Repairable Fountain codes," in 2012 IEEE International Symposium on Information Theory Proceedings (ISIT), 2012, pp. 1752–1756.

[30] K. Kasai, D. Declercq, and K. Sakaniwa, "Fountain Coding via Multiplicatively Repeated Non-Binary LDPC Codes," IEEE Transactions on Communications, vol. 60, no. 8, pp. 2077–2083, 2012.

[31] J. du Toit and R. Wolhuter, "A Practical Implementation of Fountain Codes over WiMAX Networks with an Optimised Probabilistic Degree Distribution," presented at the ICSNC 2011, The Sixth International Conference on Systems and Networks Communications, 2011, pp. 32–37.

[32] F. Xie and X. Lin, "Design of Fountain Codes with Differential Evolution," in 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp. 1–4.

[33] L. Xuehong, X. Fei, and L. Jiaru, "Designing of Fountain Codes in Cooperative Relay Systems," in 2010 Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010, vol. 2, pp. 146–149.

[34] D. J. C. MacKay, "Fountain codes," Communications, IEE Proceedings-, vol. 152, no. 6, pp. 1062–1068, 2005.

[35] V. Palma, E. Mammi, A. M. Vegni, and A. Neri, "A Fountain Codes-Based Data Dissemination Technique in Vehicular Ad-Hoc Networks," in 2011 11th International Conference on ITS Telecommunications (ITST), 2011, pp. 750–755.

[36] H. Chen, R. Maunder, and L. Hanzo, "Fountain-Code Aided File Transfer in 802.11 WLANs," in Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th, 2009, pp. 1–5.

[37] A. Ksentini and T. Chahed, "Extending the Ad Hoc Horizon in Dense 802.11 Networks Using Fountain Codes," in Fourth International Conference on Systems and Networks Communications, 2009. ICSNC '09, 2009, pp. 63–67.

[38] Y. Dai, P. Yang, G. Chen, and J. Wu, "CFP: Integration of Fountain Codes and Optimal Probabilistic Forwarding in DTNs," in 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), 2010, pp. 1–5.

[39] Z. Zhou, H. Mo, Y. Zhu, Z. Peng, J. Huang, and J.-H. Cui, "Fountain code based Adaptive multi-hop Reliable data transfer for underwater acoustic networks," in 2012 IEEE International Conference on Communications (ICC), 2012, pp. 6396–6400.

[40] M. Mitzenmacher, "Digital Fountains: A Survey and Look Forward," in IEEE Information Theory Workshop, 2004, 2004, pp. 271–276.

[41] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in Twelfth Workshop on Parallel and Distributed Simulation, 1998. PADS 98. Proceedings, 1998, pp. 154–161.

[42] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility - an Overview," presented at the SIMUL 2011, The Third International Conference on Advances in System Simulation, 2011, pp. 55–60.