Bundles Fragmentation in Vehicular Delay-Tolerant Networks

Naércio Magaia, Paulo Rogério Pereira, Augusto Casaca, Joel J. P. C. Rodrigues, João A. Dias, João N. Isento, Cristina Cervelló-Pastor, Javier Gallego

***Abstract*—Vehicular Delay-Tolerant Networks use the delay-tolerant architecture and protocols to overcome the disruptions in network connectivity. These concepts help in cases where the network is sparse or with large variations in density or there is no end-to-end connectivity, by providing a communications solution for non real-time applications. This paper presents data fragmentation techniques to optimize the efficiency of data delivery for the case of the short node contacts that characterize vehicle networks. The techniques were tested in a laboratory environment with portable digital assistants and Lego Mindstorm NXT robotic cars. If no fragmentation is used, only small messages are successfully transferred. Proactive fragmentation fragments messages to a predefined size in the source node. Reactive fragmentation adjusts the fragment sizes to the real duration of the contact when it is broken. Reactive fragmentation showed a good efficiency in adapting the fragmentation in real time to the contact duration. Proactive fragmentation can perform slightly better if the fragment sizes are carefully chosen as it requires less processing. As this choice is difficult, reactive fragmentation is more practical to use.**

***Index Terms*—Vehicular Delay-Tolerant Networks, Delay/Disruption-Tolerant Networks, Vehicular Ad Hoc Networks, Fragmentation.**

I. INTRODUCTION

DELAY-TOLERANT Networks (DTNs) [1] are networks that enable communication where connectivity issues like sparse and intermittent connectivity, long and variable delay, high latency, high error rates, highly asymmetric data rate, and even no end-to-end connectivity exist.

The DTN Research Group (DTNRG) [2], which was chartered as part of the Internet Research Task Force (IRTF), has proposed an architecture [3] and a communication protocol [4] (the Bundle Protocol) for DTNs.

Instead of working end-to-end, in DTNs, a message-oriented overlay layer called "Bundle Layer" employs a store, carry and forward message switching paradigm that moves messages from node to node, along a path that eventually reaches the destination. The idea is to "bundle" together all the information required for a transaction, minimizing the number of round-trip exchanges, which is useful when the round-trip time is very large.

Vehicular Delay-Tolerant Networks (VDTNs) are DTNs where vehicles communicate with each other in order to disseminate messages. Some of the potential applications for these networks are the following: road safety, traffic monitoring, driving assistance, entertainment, advertisements, delivering connectivity to rural/remote communities or catastrophe-hit areas, and gathering information collected by vehicles such as road pavement defects.

We proposed [5] a layered architecture for VDTNs (Fig. 1), where the bundle layer is placed below the network layer instead of over the transport layer as in the DTN architecture. The objective is to route large size messages instead of small size IP packets. This results in fewer packets processing and routing decisions, which can be translated to less complexity, lower cost and energy savings.

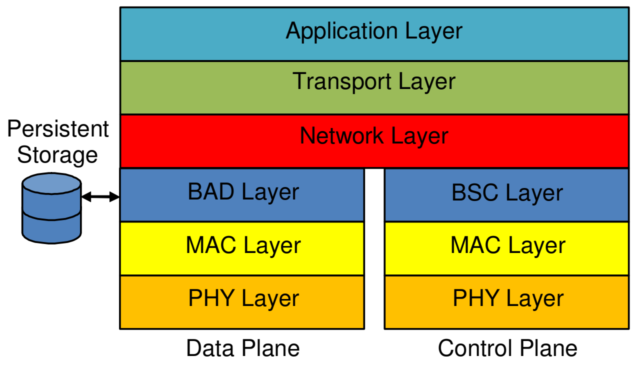


Fig. 1. VDTN protocol architecture.

The VDTN architecture uses out-of-band signaling, based on the separation of the control plane and data plane. The Bundle Aggregation and De-aggregation (BAD) layer aggregates incoming IP messages into bundle messages that

Manuscript received March, 2011. Part of this work has been supported by the *Instituto de Telecomunicações*, Next Generation Networks and Applications Group (NetGNA), Portugal, in the framework of the VDTN@Lab Project, by the Euro-NF Network of Excellence of the Seventh Framework Programme of EU, in the framework of the Specific Joint Research Project VDTN, as well as by FCT (INESC-ID multiannual funding) through the PIDDAC program funds.

N. Magaia, P. Pereira and A. Casaca are with INESC-ID, Instituto Superior Técnico, Technical University of Lisbon, Rua Alves Redol, nº 9, 1000-029 Lisboa, Portugal. (phone: +351-213100345; fax: +351-213145843; e-mails: naercio.magaia@ist.utl.pt, {prbp, augusto.casaca}@inesc.pt).

J. Rodrigues, J. Dias and J. Isento are with Instituto de Telecomunicações, NetGNA Group, University of Beira Interior, Covilhã, Portugal. (e-mails: joeljr@ieee.org, {joao.dias, joao.isento}@it.ubi.pt).

C. Cervelló-Pastor and J. Gallego are with the Dept. of Telematics Engineering, Universitat Politècnica de Catalunya (UPC), Esteve Terradas, 7, Castelldefels, Spain. (e-mails: {cristina, javier.gallego}@entel.upc.edu).

are transferred in the data plane and de-aggregated at the destination. The Bundle Signaling Control (BSC) layer provides a signaling protocol for use at the connection setup phase. The nodes exchange control information to discover each other's characteristics and prepare the data transfer to occur in the data plane. This layer also includes routing algorithms. This separation of the control and data planes comes from the Optical Burst Switching techniques.

Vehicular networks are characterized by scarce transmission opportunities and intermittent connectivity, particularly in rural or mountainous areas. A study [6] shows that the duration of contacts between cars using IEEE 802.11g crossing at 20 Km/h is about 40 s, at 40 Km/h is about 15 s and at 60 Km/h (relative speed 120 Km/h) is about 11 s. If TCP is used at 60 Km/h, the goodput is very low (average of 80 KB) and in 4 out of 10 experiments no data was transferred at all. UDP gives better results, with about 2 MB transferred in a contact at 60 Km/h.

As contacts between vehicles are short, it is important to study fragmentation methods for VDTNs. If the contact is not long enough to transfer an entire bundle message and fragmentation is not used, the incomplete message is deleted. This results in a waste of transmission resources and a decrease on the message delivery ratio or an increase in the message delivery time.

The remaining of the paper is structured as follows. Section II presents an overview of fragmentation in DTNs. The following section details on the proposed fragmentation mechanisms. Section IV presents the experimental evaluation of the mechanisms. Section V presents conclusions and further work topics.

II. Related Work

Fragmentation in IP networks has been considered harmful [7]. It can lead to poor performance or complete communication failure. The loss of fragments requires the higher layer protocol to retransmit all of the data in the original datagram. Fragmentation also introduces additional headers, requiring additional bandwidth, additional processing at intermediate nodes, and complex reassembly mechanisms at the receiver. For this reason, IPv6 [8] disallows fragmentation inside the network, requiring that fragmentation is done in the source node to comply with the path's maximum transmission unit (MTU) to the destination.

However, large application messages can benefit from fragmentation when connections are disrupted, a normal circumstance in DTN. An example is the HTTP protocol that allows byte ranges of objects to be individually retrieved [9], enabling partial file retrievals to be resumed without restarting the transfer from the beginning.

Two types of fragmentation are foreseen in the DTN architecture [3]: proactive and reactive. In proactive fragmentation, a DTN node may divide a block of application data into multiple smaller blocks and transmit each block as an independent bundle. This approach is called proactive fragmentation because it is used primarily when the contact duration is known (or predicted) in advance. In the reactive fragmentation, the DTN nodes in communication may fragment a bundle cooperatively when a bundle is only partially transferred; the receiving bundle layer modifies the incoming bundle to indicate the data received is a fragment and forwards it normally; the sender node truncates the successfully transferred part from the original bundle and keeps the undelivered fragment for a subsequent contact. This approach is called reactive fragmentation because the fragmentation process occurs after an attempted transmission has taken place. A variant of reactive fragmentation does not allow arbitrary sizes of fragments, creating fragments only at pre-defined boundaries of the message, as defined by the source. This variant is often called toilet paper approach [10], which allows fragment authentication.

The bundle protocol [4] defines how fragments are created, processed and reassembled. However, little is said on how and when to use fragmentation.

Proactive fragmentation is particularly adequate for satellite networks where the connectivity periods are known. An example is provided in [11], where very large files transfers are proactively fragmented at the source. In addition, connection problems can lead to subsequent reactive fragmentation of the initial fragments. Proactive and reactive fragmentation are supported by the DTN2 reference implementation [12], but not for all lower layer protocols.

As in vehicle networks, contact durations are not know a priori, the only way to use proactive fragmentation is to fragment the message in fragments of a pre-determined size in the first node.

The work in [13] explores the use of node localization in vehicles, as provided by a Global Positioning System (GPS), to estimate the duration of contacts between vehicles and schedule the message transmission order to prevent incomplete transmissions. But no fragmentation mechanism is explored.

The most complete study of message fragmentation in DTN through simulation is provided in [14]. It concludes that the effects of fragmentation are largely independent of the routing protocol. Proactive fragmentation can reduce the performance, as fragments can be spread across multiple paths and the loss of one compromises the whole message. Reactive fragmentation can effectively increase overall connectivity, improving the delivery ratio, while the latency remains about the same. The reactive fragmentation with predefined fragment boundaries (”toilet paper approach”) shows slightly better performance as it avoids small fragments. However, this study is only based on simulations. In this paper, we study fragmentation mechanisms based on a real testbed environment.

III. VDTN Fragmentation

In a real network, a synchronization problem between adjacent nodes may arise when a connection is broken. If the sending node has successfully passed the data to the operating system for transmission and the connection is broken, the operating system drops the data without warning. This might result in the sender thinking that the data was successfully transmitted, while it was not received by the next node in the path to the destination. This problem may be solved by a convergence layer adapter that performs the adaptation to the lower protocol layers. This convergence layer would provide an acknowledgement that the data was successfully transferred. So, any synchronization problem might result in a subsequent retransmission of the data and a possible overlapping of data that has to be solved by the final destination. Without the acknowledgement, data loss might occur.

Fig. 2 shows the implementation architecture where a bundle forwarder interacts with storage, routing decisions, a bundle fragmentation and reassembly layer and a convergence layer adapter to interface to a delivery protocol. Note that this fragmentation and reassembly layer is a sub-layer of the bundle layer that deals with fragmentation and reassembly of bundles internally to the network. The bundle layer of Fig. 1 deals with building bundles with the application data at the communication end points (source and destination).

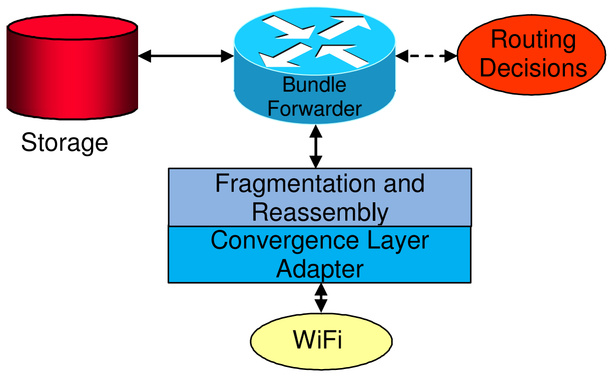


Fig. 2. Implementation architecture.

Naturally, the synchronization problem does not occur if the routing protocol generates an arbitrary number of copies of the messages, such as the Epidemic routing [15] that floods the messages to all nodes. In this case, whether the data was transferred or not, the next contact will provide another opportunity to transfer the data that had not yet been transferred. So, the acknowledgement in the convergence layer adapter is optional in this case. However, for routing protocols that keep a single copy of the messages, the acknowledgment is necessary so that the fragmentation is independent of the routing protocol. Examples of such protocols are Direct Delivery routing, in which the node originating a message carries it until it meets its final destination, and First Contact routing, where the nodes forward messages to the first node they encounter, which results in a "random walk" search for the destination node.

For comparison purposes, we will test communications over a VDTN in three scenarios: i) no fragmentation; ii) proactive fragmentation; iii) reactive fragmentation.

Without fragmentation, when a bundle is only partially transmitted, it is deleted. This prevents the transmission of messages larger than the size that can be transferred in a contact between vehicles. As contacts may be rather short, this is a serious limitation, as will be demonstrated.

For proactive fragmentation, the bundles are fragmented, when entering the VDTN, into fragments of a given size for transmission. The bundle is reassembled only at the destination when all fragments are received. As each fragment has its own headers, fragmenting into small fragments introduces an increased transmission overhead and an increased processing overhead. If the fragments are large, more data can be lost when the contact between nodes is broken. So, there is a compromise in setting the length of fragments. It is very difficult to accurately predict the duration of contacts between vehicles, as vehicles may change direction without warning. Using GPS, information about the road where the car is and a received signal strength indicator (RSSI) may help estimating the duration of the contact. To be more general, we assumed no such information was available and tested with different fixed fragment sizes. Messages are lost if any of the fragments does not reach the destination. Fragments may be lost if: i) their time to live (TTL) expires, maybe because they took a very long path; ii) memory is exhausted at some node due to congestion; iii) a node leaves the network, maybe because the vehicle is turned off.

Reactive fragmentation requires that during a contact, the nodes can determine which part of the data has been successfully transferred and which has not. The receiver creates a fragment with all the data correctly received. The sender keeps a fragment with all the untransferred data. As the radio transmission over a Medium Access Control (MAC) layer is packet oriented, we do not consider the possibility of fragments of arbitrary size, but only multiples of a certain block size.

To illustrate the fragmentation operation, we consider an example where the bundle message has 10KB data, the contact allows the transmission of approximately 9.5KB and the blocks transferred by the convergence layer adapter are of 4KB. Fig. 3 shows the operation of the three fragmentation scenarios (not to scale).

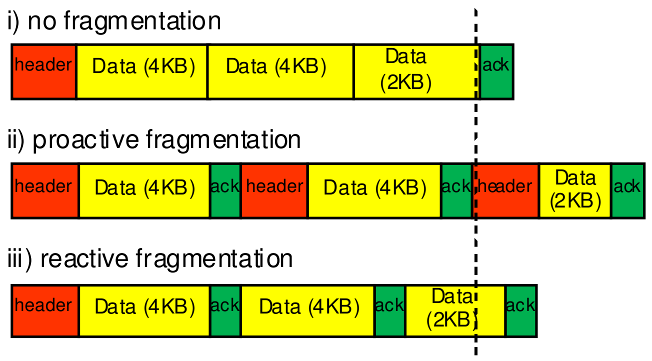


Fig. 3. Fragmentation example.

For this example, when no fragmentation is used, the bundle is not entirely transferred, so it is discarded. When proactive fragmentation is used, the two first 4KB fragments are transferred and the last 2KB fragment is not. When reactive fragmentation is used, the bundle is fragmented into an 8KB fragment successfully transferred and a 2KB fragment not transferred. In this example, we assumed that no synchronization problem occurred. If it did occur, some overlapping data might be present when reassembling the bundle at the destination. We also neglected the processing times.

In addition to the increase in data transferred when fragmentation is used, the proactive fragmentation also has an increase in signaling to announce more fragments available for transfer during a contact. In the example of Fig. 3, the signaling required for proactive fragmentation would be the triple of the no fragmentation or the reactive fragmentation cases.

IV. Experimental Evaluation

For assessing the performance of the fragmentation mechanisms, we used a mobile node going back and forth crossing a fixed node. The mobile node is a Lego Mindstorm NXT robotic car coupled with a Samsung OMNIA II GT-i8000 Personal Digital Assistant (PDA) running Windows Mobile version 6.5.3 Professional. The fixed node is a laptop running Windows 7. The applications were developed using the C# language in a Microsoft Visual Studio environment.

All nodes use Bluetooth 2.0 (3 Mbit/s) for out-of-band signaling and IEEE 802.11b (11 Mbit/s) connections for data transfers, and storage capacities to allow VDTN data communications. The routing protocol is Epidemic, where the nodes try to transfer the messages to all contacts until they expire or reach the destination. The message TTL was set longer than the experiments so that the messages have an opportunity to be delivered to the destination without expiring. No storage limitation was imposed on the nodes, so that no message is deleted due to congestion.

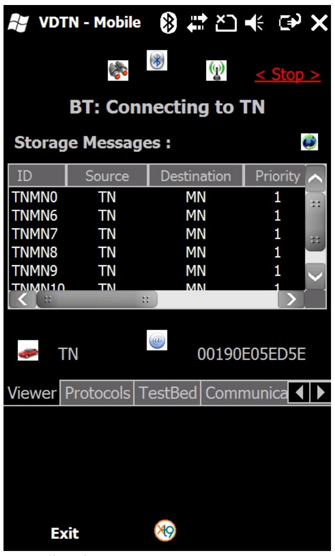
The data plane convergence layer is using the User Datagram Protocol (UDP) over IP over IEEE 802.11b. In the PDA used, the maximum datagram length is 32 KB, so the maximum transfer block size used was 32 KB.

The contact times vary from 9 to 21 seconds (average 15 seconds, standard deviation 3.5 seconds), which according to [6] corresponds to vehicles crossing at a relative speed of approximately 80 Km/h. The interval between successive contacts varies from 1 to 27 seconds (average 9 seconds, standard deviation 7 seconds).

For each situation tested, the corresponding test was repeated 20 times and the results averaged. For some situations, the tests were repeated 20 additional times to improve the confidence level of the results.

When a contact is established, the nodes inform each other, through signaling, which bundles they have and decide which bundles are to be transferred. Then, they transfer them through the data link.

Fig. 4 and Fig. 5 show, respectively, the mobile node and terminal node application screens. Several messages from the terminal node (TN) to the mobile node (MN) are shown with the corresponding details for debugging.



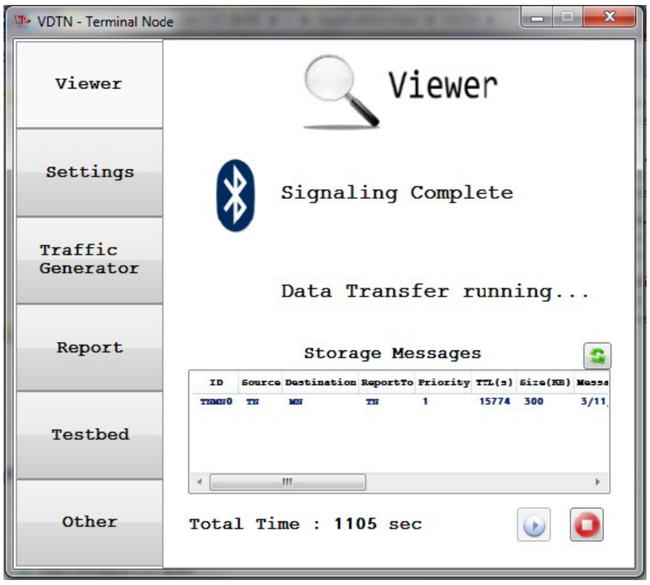
BT: Connecting to TN

Storage Messages :

ID	Source	Destination	Priority
TNMN0	TN	MN	1
TNMN6	TN	MN	1
TNMN7	TN	MN	1
TNMN8	TN	MN	1
TNMN9	TN	MN	1
TNMN10	TN	MN	1

TN 00190E05ED5E

Fig. 4. Mobile node application screen.



Viewer

Signaling Complete

Data Transfer running...

Storage Messages

ID	Source	Destination	ReportTo	Priority	TTL(s)	Size(KB)	Messa
TNMN0	TN	MN	TN	1	15774	300	3/11,

Total Time : 1105 sec

Fig. 5. Terminal node application screen.

A first interesting result is the average number of contacts necessary to transfer a message. Fig. 6 shows the experimental results for this measurement, using a 32 KB block size for the different scenarios. When fragmentation is not used, the message is only transferred if the contact is long enough, so when the message size increases, the number of contacts long enough to transfer the entire message is dramatically reduced. No value for 900 KB messages without fragmentation is presented, as this message size is too long for successful transmissions in our scenario. In contrast, a 900 KB message does not require more than an average 1.4 contacts to be transferred with either fragmentation method. This result shows the importance of fragmentation methods when contacts are short, as in vehicular networks. The difference between the two fragmentation methods is negligible in this aspect.

Another important aspect is the effect of varying the block size. Fig. 7 shows the average message delivery latency when the block size is 4 KB, while Fig. 8 shows the same results when the block size is 8 KB and Fig. 9 for a block size of 32 KB. When no fragmentation is used, it is very difficult to transfer an entire 600 KB message in our test scenario, so only the value for 32 KB blocks is shown. Again, as it is almost impossible to transfer a 900 KB message without fragmentation, the corresponding results are not shown.

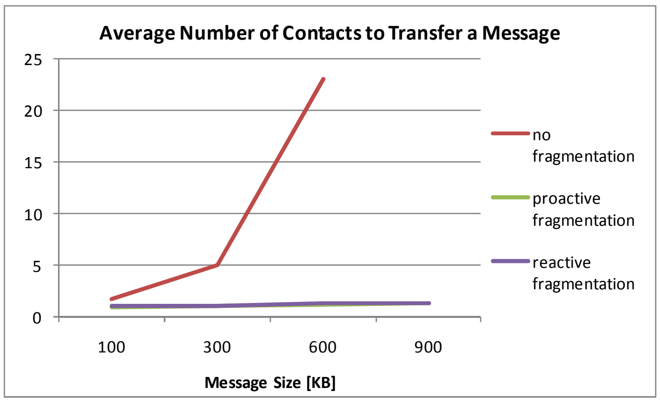


Fig. 6. Average number of contacts to transfer a message as a function of the message size.

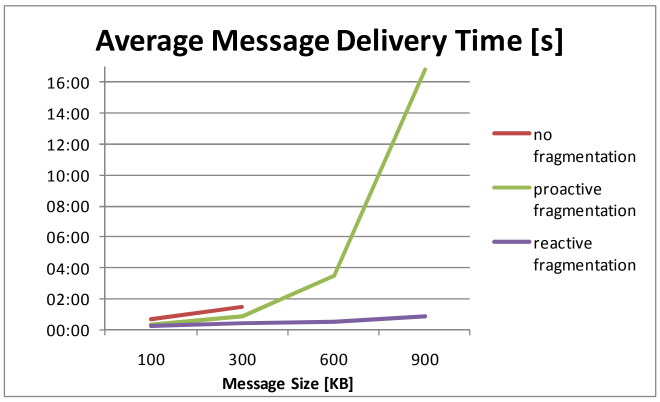


Fig. 7. Average message delivery latency for a block size of 4KB.

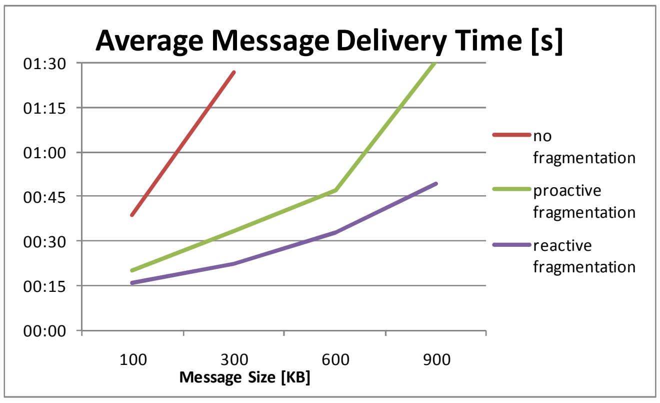


Fig. 8. Average message delivery latency for a block size of 8KB.

For a small block size, such as 4 KB, the proactive fragmentation has a significant cost in terms of additional signaling to inform the peer entity, during a contact, of the numerous (4 KB) fragments available for transfer and in terms of additional headers for each fragment (12% to 17%, as listed in Table I). So, although the messages are successfully transferred, the delivery latency is strongly affected, particularly for larger message sizes. However, we note that fragmentation is worth being used, as even for messages as small as 100 KB, the waste of resources for incomplete transmissions when no fragmentation is used exceeds the overhead incurred with fragmentation. Reactive fragmentation performs much better for the different message sizes, without a degradation effect for larger messages.

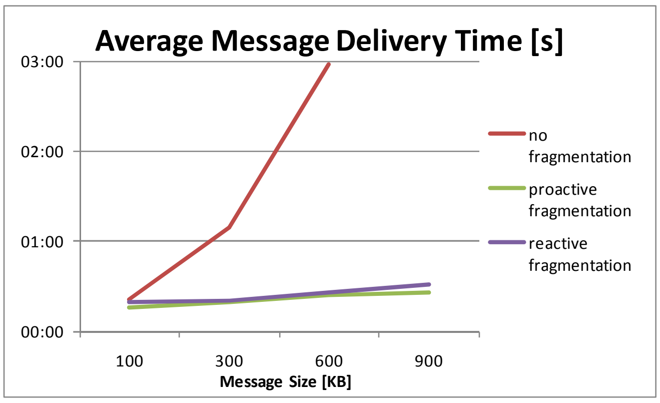


Fig. 9. Average message delivery latency for a block size of 32KB.

TABLE I
QUANTITY OF INFORMATION FOR PROACTIVE FRAGMENTATION FOR DIFFERENT MESSAGE SIZES AND BLOCK SIZES

Message Size / Block Size	100 KB	300 KB	600 KB	900 KB
4 KB	117 KB	347 KB	693 KB	1010 KB
8 KB	108 KB	322 KB	642 KB	962 KB
32 KB	103 KB	306 KB	611 KB	915 KB

For an 8 KB block size, the fragmentation overhead is reduced. However, the proactive fragmentation still performs 25-85% worse than reactive fragmentation. Both fragmentation methods greatly outperform the no fragmentation case.

For a larger block size, such as 32 KB, the overhead incurred by proactive fragmentation has little significance (2% to 3% more headers, as listed in Table I), so the results of proactive and reactive fragmentation are very similar. The differences in the results shown (0-5 seconds) are smaller than their 95% confidence intervals, which are 1-5 seconds for the proactive fragmentation and 4-9 seconds for the reactive fragmentation, depending on the message size. Although proactive fragmentation requires the transmission of more information, it performs slightly better than reactive fragmentation for a 32 KB block size as it requires less processing during the contacts. This is because fragments are generated in advance in the source node when the bundle message is created. On the other hand, reactive fragmentation requires some additional processing during the contact to determine if any part of a message is missing, possibly causing the transfer of some overlapping part. When comparing the results without fragmentation for 4 KB and 32 KB block sizes, we find that the 32 KB blocks are more efficient. The difference is that with 32 KB blocks, less system calls are generated to transfer the same application data, which leads us to conclude that there is a considerable additional overhead in processing the data in the application layer as compared with processing it in the operating system, particularly for systems with limited processing capabilities such as PDAs. This conclusion also justifies that reactive fragmentation is slightly less efficient than proactive fragmentation for 32 KB blocks, as it requires more processing. When a contact is broken, the impact of interrupting the data transfer of a 32 KB block as compared with a 4 KB block is negligible as compared with the corresponding processing overhead, so it is more efficient to use larger blocks.

However, an important advantage of reactive fragmentation is that it adapts to the contact duration in real-time, which is particularly important when the contact duration is not know in advance. This is confirmed by the message delivery times of the reactive fragmentation being almost the same for the different block sizes. Using a 4 KB block size with reactive fragmentation results only in an average 20% increase in the delivery time as compared with the 32 KB block size case. In contrast, reducing the block size from 32 KB to 4 KB with proactive fragmentation can result in a 39 times increase of the delivery time for a 900 KB message.

As for memory occupancy, proactive fragmentation requires some additional memory to store the fragments as new headers are necessary as compared to the situation without fragmentation. The required memory is as listed in Table I. Reactive fragmentation only requires more memory as compared to the situation without fragmentation if fragments are really created, so the increase in memory use is negligible.

V. Conclusion

By applying DTN concepts to vehicular networks, new challenging situations typical of vehicular networks may be overcome, such as sparse and intermittent connectivity, variable delays, high error rates and inexistence of an end-to-end path.

The use of fragmentation allows the transfer of messages larger than the typically short contacts between vehicles, resulting in an increased delivery ratio and decreased delivery latency.

Reactive fragmentation adapts the fragments size dynamically to the duration of the contact, resulting in an efficient use of the contact opportunities. In contrast, proactive fragmentation requires an a priori definition of the fragment size, which is difficult to adjust to the duration of the contact opportunity in a VDTN. If the fragments are set too short, proactive fragmentation is inefficient.

However, reactive fragmentation requires more processing than proactive fragmentation so, if the fragments of the proactive fragmentation are well adjusted to the contact opportunity, the latter may perform slightly better than reactive fragmentation.

Some future work possibilities are testing more complex VDTN scenarios, with more nodes and mixes of different message sizes; using contact duration prediction mechanisms to combine scheduling and fragmentation of bundles to optimize the data transfer phase.

References

[1] Kevin Fall and Stephen Farrell, "DTN: An Architectural Retrospective", *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 828-836, June 2008.

[2] Delay Tolerant Networking Research Group. http://www.dtnrg.org/wiki

[3] V. Cerf et al., "Delay Tolerant Network Architecture", IETF, RFC 4838, April 2007.

[4] K. Scott and S. Burleigh, "Bundle Protocol Specification", IETF, RFC 5050, November 2007.

[5] V. Soares, F. Farahmand and J. Rodrigues, "A Layered Architecture for Vehicular Delay-Tolerant Networks", *IEEE Symposium on Computers and Communications (ISCC 2009)*, Sousse, Tunisia, July 5-8.

[6] M. Rubinstein *et al*, "Measuring the Capacity of In-Car to In-Car Vehicular Networks", *IEEE Communications Magazine*, vol. 47, no. 11, pp. 128-136, November 2009.

[7] C. Kent, J. Mogul, "Fragmentation considered harmful", ACM SIGCOMM Computer Communication Review, Vol. 25, Issue 1, Jan. 1995.

[8] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", IETF RFC 2460, December 1998.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1". IETF RFC 2616, June 1999.

[10] S. Farrell, S. F. Symington, H. Weiss, P. Lovell, "Delay-Tolerant Networking Security Overview", Internet Draft draft-irtf-dtnrg-sec-overview-06, March 2009.

[11] W. Ivancic, P. Paulsen, D. Stewart, J. Taylor, S. Lynch, J. Heberle, J. Northam, C. Jackson, L. Wood, "Large File Transfers from Space using Multiple Ground Terminals and Delay-Tolerant Networking", *IEEE Global Communications Conference (Globecom 2010)*, Miami, USA, December 2010.

[12] DTN Code. http://www.dtnrg.org/wiki/Code

[13] Vasco Soares, Joel Rodrigues, Farid Farahmand, and Mieso Denko, "Exploiting Node Localization for Performance Improvement of Vehicular Delay-Tolerant Networks", *IEEE International Conference on Communications (ICC 2010)*, Cape Town, South Africa, May 23-27, 2010.

[14] M. Pitkänen, A. Keränen, J. Ott, "Message Fragmentation in Opportunistic DTNs", *Second WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC)*, 2008.

[15] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks", Tech. Rep. CS-200006, Department of Computer Science, Duke University, Durham, NC, 2000.