Effective Buffer and Storage Management in DTN Nodes

Stylianos Dimitriou, and Vassilis Tsaoussidis
Dept. of Electrical and Computer Engineering
Democritus University of Thrace
Xanthi, Greece
{sdimitr, vtsaousi}@ee.duth.gr

***Abstract*—Current wired networks have been developed on the basis of the AIMD principle, which offers increased performance and fairness. Nevertheless, there is a vast spectrum of networks, from deep space to wireless sensor networks, where TCP fails to operate, as it results in frequent timeouts and intermittent or no connectivity at all. The DTN (Delay/Disruption Tolerant Networking) architecture and particularly the Bundle protocol, offers solutions in networks where low connectivity, big or variable delays and increased error rates are prevalent. Using persistent storage, a DTN node is able to overcome such limitations; the data can be transmitted several times until it is successfully received by the next hop. However, since the conditions in a DTN network might vary, from time to time we may experience high connectivity and low delays. In such cases, data does not need to be transferred to persistent storage, as it will be probably be transmitted shortly and this transfer from buffer to persistent storage and back to buffer, will delay the transmission. In this paper we propose a system which implements a mechanism that aims to minimize packet transfers between buffers and persistent storage and hence accelerate transmissions.**

Keywords-component; DTN; Bundle Protocol; Buffer Management

I. INTRODUCTION

The Bundle protocol [3] aims to resolve issues in DTN networks, such as intermittent connectivity, long or variable delays and high error rates. Examples of DTN networks are MANET, vehicular networks and deep-space networks. In most of these cases, there is not a continuous end-to-end path between the sender and the receiver; the receiver might be available after the sender has been disconnected or an intermediate routing node may be moving constantly between the always connected, albeit remote communicating ends. Using persistent storage, the Bundle protocol manages to resolve most of these issues. Instead of storing packets in a temporary buffer, a DTN node stores packets permanently in a local storage unit. Data will be transmitted when the next hop is available and may be retransmitted several times, if the transmission fails. Persistent storage, however, has a significant drawback: it increases the communication overhead as it adds a processing delay which results by moving the data from buffer to persistent storage and back to buffer. Although most of the times this delay is negligible, there are cases where two nodes may be engaged in continuous low-delay transmissions. In such cases, persistent storage increases unnecessarily the connection time.

In this work, we investigate the potential of maintaining data-to-transmit in buffers, instead of moving them to persistent storage. We model a system where packets arrive from different flows. Based on the frequency of connections to the next hop, we do not store all packets in persistent storage, but we maintain some of them in buffers. In this way, we aim to decrease service time and processing delay and hence increase performance. Since our proposal consists of work-in-progress, we do not include any evaluation to prove our claims.

The rest of the paper is organized as follows. In Section 2 we give a brief background on DTN architecture and protocols. In section 3 we present the proposed model and the buffer management algorithm and in section 4 we set the framework for future work.

II. BACKGROUND

The structure and protocols of Internet have been evolved around some characteristics that are ubiquitous in wired networks: available end-to-end path, small delays and low error-rates. The need to propose different protocols, in order to implement Delay Tolerant Networking [1], was apparent from the beginning, since classical network protocols performed poorly in DTN networks. In order to be able to interconnect multiple regional DTN networks with themselves, as well with the Internet, an overlay layer was proposed, namely the Bundle layer, where the Bundle protocol is implemented. Contrary to TCP/IP networks where, in case of an unavailable link, the packets are discarded, the Bundle protocol specifies that traffic should be transferred in a Store-and-Forward fashion; packets arriving at the node should be saved in a storage unit, until the next hop becomes available. In order to ensure the integrity of data, a Custody Transfer mechanism may be used. A node that accepts data transferred with custody transfer should preserve the data until transmitted to the next hop and not drop them, even in case of a full storage unit. Several modes exist concerning the bundle transmissions, involving acknowledgments and notification. Nevertheless, there are cases where we seek to decrease the complexity of the system, in order to decrease the latency.

9781-4244-3941-6/09/$25.00 ©2009 IEEE

III. SYSTEM MODEL

We will present a basic model which we will use to apply our algorithm. Since our work is still in-progress, we will try to be as abstract as possible. Further details, as well as parameter values will be determined in the future.

A. Modeling Arrivals and Departures

We consider a node which acts an intermediate to several flows. That is, packets of several senders enter the node at various instances. The flows can be identified either by the IP addresses of the sender and receiver, if IP is used, or by the endpoint ID. Since we assume network mobility, the node may accept packets either from other routing nodes or directly from senders. In the first case, the packets arrive back-to-back with constant interarrival times, and in the second case, in a stochastic manner. The node then has to keep them until a connection opportunity occurs, or until its storage space is full. Each node has a buffer and a persistent storage unit, both limited. The buffer consists of two queues; a low-delay traffic (LDT) queue and a high-delay traffic (HDT) queue. Fig. 1 shows the proposed model. Note that at this point we do not determine the sizes of the two queues. Their sizes can be either defined at the beginning, or vary as the correlation of traffic changes. Custody transfers are also stored in persistent storage.

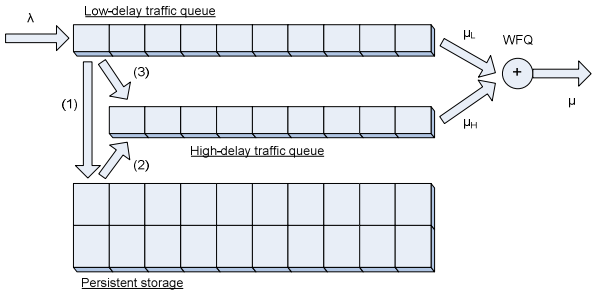

Figure 1. Proposed Model

The aggregate incoming rate is noted as λ. When a packet is accepted by the node, the node will decide if it belongs to a low-delay flow or to a high-delay flow (we will describe shortly how low- and high-delay flows are defined). If the packet belongs to a low-delay flow, it will be buffered in the LDT buffer, where it will remain until routed or until the buffer is full. In case the LDT buffer is full, older traffic will be moved to persistent storage. If the packet belongs to a high-delay flow, it will be moved immediately to persistent storage, unless there is a communication opportunity. In this case it will be moved to the HDT buffer. Packets that exist in persistent storage are allowed to move to the HDT buffer, only when the next hop for the corresponding flow becomes available. The total service rate μ results by multiplexing the output of the two queues with a Weighted Fair Queuing [2] multiplexer. Details of the WFQ mechanism are still an open issue.

Summarizing, moving packets from buffer to storage and back, is allowed only in the following three cases:

1) From LDT buffer to persistent storage. Packets that can move in this direction are either new packets that belong to a high-delay flow, or old packets that belong to a low-delay flow and the LDT buffer is full.
2) From persistent storage to HDT buffer. Packets that can move in this direction are packets that were previously in persistent storage and currently they have a communication opportunity.
3) From LDT buffer to HDT buffer upon packets arrival. Packets belonging to a high-delay flow, will move to HDT buffer if there is currently a communication opportunity with a next hop.

There is, however, a very rare case where packets would need to move from HDT buffer to persistent storage. This would happen when there is a communication opportunity, which lasts less time than the time required to move the packets from persistent storage to HDT buffer and commence transmission. Packets which were transferred to HDT buffer would have nowhere to go and should be moved back to storage. Nevertheless, this is a special case that is unlikely to happen, given the high processing capabilities of current computers.

B. Distinguishing Low- and High-Delay Traffic

In order to detect LDT, that is, traffic that we expect to have a smaller delay and generally faster forwarding, we record the time a packet enters the node (t_{in}) and the time it leaves it (t_{out}). The difference of these two times ($diff_{sample}$) equals with the storage delay of the packet, i.e. the time period where the packet remains in the node, either in buffer or persistent storage. For each flow i, we calculate the weighted moving average of the storage delays $diff_i$. The average value of all flows' storage delays gives the average storage delay of the system ($Diff$). Flows that have less average storage delay than the total average, are characterized as low-delay flows, whereas the rest are characterized as high-delay flows. Assuming N flows passing through the node, we have:

$$diff_{sample} = t_{out} - t_{in} \quad (1)$$

$$diff_i = 0.9 \cdot diff_i + 0.1 \cdot diff_{sample} \quad (2)$$

$$Diff = \frac{\sum_{i=1}^{N} diff_i}{N} \quad (3)$$

Applying Eq. 1 to every packet and Eq. 2 to every flow, we result with the average $Diff$ value (Eq. 3).

IV. FUTURE WORK

In this paper we described a buffer management scheme which can be used to achieve less processing delays in a DTN network. Our work is in-progress and thus, has several open issues. Our future steps include:

1) Determine the sizes of LDT and HDT queues. Should they be statically defined, or it is preferable to obtain values depending on the network load?
2) Review the WFQ scheme used to multiplex the outputs of the two queues. This will also define the bandwidth

that each type of traffic will possess. Failure to do so and we will end up in underutilization.

3) Evaluate the proposed algorithm, both theoretically and experimentally. The results will be used to further calibrate the model, where necessary.

REFERENCES

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

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," Journal of Internetworking Research and Experience, pages 3-26, October 1990.

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